
HOBBY COMPUTER
Sonderheft der
ELO
Funkschau
Elektronik
Prei
DM 19,
Grundlagen für
den Anfänger
Heiße Themen
für den Profi
Radio Shack
TRS-80

VORWORT

Im Franzis-Verlag sind bisher zwei Sonderhefte über Mikroprozessoren erschienen, die auf die industrielle Praxis zugeschnitten sind.

Seither ist die Computertechnik – vorangetrieben durch die ständig fallenden Preise – insbesondere in den USA in den privaten Bereich vorgedrungen. In Europa zeichnet sich eine ähnliche Entwicklung ab. Der Laie steht aber zunächst ratlos vor dem unglaublich leistungsfähigen Werkzeug, das ihm da angeboten wird. Diese Informationslücke durch die Vermittlung der wichtigsten Grundlagen zu schließen, ist ein Anliegen des vorliegenden Sonderheftes.

Ein zweites Anliegen ist es, dem Fortgeschrittenen Ideen, Anwendungen und noch mehr Wissen zu vermitteln. Naturgemäß berühren sich dabei die Bereiche ,,Hobby'' und ,,Berufspraxis''. Das zeigt auch die Zusammensetzung der Autoren: Sie sind überwiegend Fachleute, die auch beruflich mit der Sache zu tun haben, beschäftigen sich aber ausnahmslos auch privat mit Computern – einfach weil es Spaß macht.

Die Redaktion



Foto: Winkler

Die Hausfrau hat im „Geschäft um die Ecke eben mal" einen Computer gekauft.

Zukunftsmusik?

Nein, nicht unbedingt! Denn schon heute kann man das Gerät auf unserem Bild – ein vollwertiger Computer – für etwa 1800 DM kaufen. Um ihn zu programmieren, braucht man keinerlei technische Kenntnisse (ebenso wenig, wie man zur Bedienung eines Taschenrechners dessen Innenleben kennen muß). Der TRS-80 ist ab S. 46 beschrieben. Ein ähnliches Gerät ist der PET, der ab S. 49 vorgestellt wird.

Wer sich dennoch für die Technik interessiert, der findet ab S. 11 und im anschließenden Beitrag eine Einführung. Die dann folgenden Systeme sind in erster Linie für denjenigen interessant, der den Lötkolben nicht ganz aus der Hand legen will – für den also, der sich mit Schaltungen befassen will und auch ein klein wenig davon versteht.

Allgemeine Hinweise

Erste technische Erläuterungen

Einstieg in die Maschinensprache

Systeme für den Schaltungstechniker

Programme in Maschinensprache

BASIC-Grundlagen

BASIC selbstgemacht

Inhalt

BASIC-Systeme für Heim-Programmierer

Programme in BASIC

S-100-Grundlagen und -System

Speicherverfahren

Allerlei Nützliches

Kurzbeiträge

Franzis-Verlag GmbH, Karlstr. 37, München.
Redaktion: Rudolf Hofer, für den Inhalt verantwortlich Hans J. Wilhelmy, beide in München.
Herstellung: Sebastian Preis.

Druck: Franzis-Druck GmbH, München. Printed in Germany. Imprimé en Allemagne.

Seit kurzem ist es auch technisch völlig Unbewanderten möglich, die Computerei zu ihrem Hobby zu machen, ohne daß das Unterfangen von vornherein zum Scheitern verurteilt wäre. Der entscheidende Schritt dazu war das Erscheinen kompletter Computer, die mit einer sehr leicht zu erlernenden Programmiersprache zu bedienen sind und die schon für runde 2000 DM verkauft werden. Knappe 1000 DM reichen für den Anfang aus, wenn man elektronisch etwas bewandert ist. Beide Möglichkeiten werden im folgenden Beitrag durchleuchtet, damit sich der Einsteiger entscheiden kann, ob die Sache für ihn interessant und erschwinglich ist.

Rudolf Hofer

Computer-Hobby: ein Wegweiser für Interessierte

Was tun Computer-Bastler mit ihren Maschinen?

Diese Frage ist gar nicht so leicht zu beantworten. Es scheint nämlich tatsächlich so, als wäre dies das größte Problem der Hobbyisten. Und so verbringt ein nicht geringer Teil von ihnen seine Zeit damit, immer bessere Hilfsprogramme zu schreiben, mit deren Hilfe man immer schneller und komfortabler programmieren kann. Zur eigentlichen Anwendung kommt man aber nie. Diesen Leuten ergeht es ähnlich wie manchen Funkamateuren, die ständig bessere Übertragungsqualität anstreben, ohne jemals etwas anderes durch den Äther zu schicken als den Rapport, wie gut man ankommt und empfängt. Trotzdem: Funken macht Spaß, und Programmieren macht Spaß! Warum soll man das ganze nicht als Denksport auffassen, der das eingetrocknete Gehirnschmalz wieder auf Vordermann bringt.

Es gibt natürlich auch andere Gruppen. Etwa solche, die ein Hobby mit einem anderen verbinden und beispielsweise die elektrische Eisenbahn steuern oder synthetische Musik erzeugen.

Im großen und ganzen sind jedoch zwei Gruppen zu unterscheiden: die „Löter" und die „Tipper". Die Löter bauen sich, wenigstens zum Teil, ihr eigenes System aus Einzelbausteinen zusammen und schließen an den Computer Schaltungen an, die andere Dinge steuern (z. B. eine Eisenbahn). Sie sind meistens gewiefte Schaltungstechniker – was auch notwendig ist. Wir wollen sie später behandeln.

1 Mikrodatentechnik

Wenden wir uns also den „Tippern" zu: Sie betreiben eine Art Mikrodatentechnik und bevorzugen einen kompletten Computer, der eingeschaltet wird und sofort betriebsbereit ist. Unbedingt notwendig sind für sie Tastatur (ähnlich der einer Schreibmaschine) und Bildschirm. Es gibt zwar viele sogenannte Einkartencomputer (Abschn. 2), die eine kleine Tastatur und Leuchtziffern-Anzeige haben, die sollten aber den „Lötern" überlassen werden, obwohl sie meist weniger als ein Drittel der eingangs erwähnten Komplettsysteme kosten. Im Endeffekt wird der Spaß aber dann doch wesentlich teurer, denn auf die Dauer gibt sich *keiner*, der im herkömmlichen Sinn Daten verarbeiten will, mit dieser Schmalspurausführung zufrieden. Die Erweiterung um ein anständiges Sichtgerät (1400 DM mindestens) bringt uns dann bereits auf den Gesamtpreis des Komplettsystems. Allerdings fehlen uns dann noch Speicher und höhere Programmiersprachen, was nochmal mit etlichen hundert Mark zu Buche schlägt.

Die Hauptanwendung der Mikrodatentechnik liegt vor allem in der Textverarbeitung und im Rechnen. Als Programmiersprache wird ausschließlich BASIC verwendet, das bei geringen mathematischen Vorkenntnissen in kurzer Zeit (wenige Tage) soweit erlernt werden kann, daß schon ganz nette Programme erstellt werden können. Die Schwierigkeitsstufe von BASIC ist etwa vergleichbar mit der Bedienung von programmierbaren Taschenrechnern. Die Möglichkeiten sind aber, wie gesagt, ungleich vielfältiger, da nicht nur Zahlen, sondern auch Texte bearbeitet werden können. So sind beispielsweise Zwiegespräche mit dem Computer möglich, der – abhängig von der Eingabe – Fragen stellt oder beantwortet. Das geht natürlich in Richtung „Spiele". Aber auch Mini-Datenbanken können auf diese Weise verwirklicht werden. Man könnte z. B. die täglichen Latein- oder Englisch-Vokabeln einspeichern und den jeweiligen deutschen Wörtern zuordnen.

Wird dann ein beliebiger deutscher Begriff eingetippt, dann antwortet der Computer mit dem entspre-

chenden englischen und umgekehrt. Dieses Beispiel verdeutlicht die Grundidee für das Auffinden eingespeicherter Daten. Natürlich kann man einem Suchbegriff auch beliebige andere Texte zuordnen – Adressen, Telefonnummern, Tagebucheintragungen, Kundendaten usw.

1.1 Kommerzielle Anwendungen setzen Peripheriegeräte voraus

Das bringt uns zu den sogenannten kommerziellen Anwendungen – nützlichen Dingen also, die man etwa für die Buchhaltung eines Geschäftes benutzen kann. Darunter fallen beispielsweise die tägliche Abrechnung der Einnahmen, die Kontrolle des Lagerbestandes, das Führen einer Statistik, mit deren Hilfe man den Bedarf kalkuliert usw. Grundsätzlich ist das natürlich mit einem Grundsystem möglich. Im allgemeinen braucht man jedoch dafür Peripheriegeräte – einen Drucker mindestens, besser noch zusätzlich ein Floppy-Disk-Laufwerk.

Um den Begriff Floppy Disk zu erklären, muß zunächst gesagt werden, daß man für jeden Computer eine externe Speichermöglichkeit braucht. Die entwickelten Programme sollen ja nicht jedesmal neu eingetippt werden, wenn das Gerät einmal ausgeschaltet war. Die billigste Speichermöglichkeit ist ein handelsüblicher Cassettenrecorder, der bei den gebräuchlichen Systemen meist mitgeliefert wird oder schon eingebaut ist. Für viele Hobby-Anwendungen reicht das völlig aus. Sollen aber große Datenmengen abgespeichert werden, aus denen man in kurzer Zeit ganz bestimmte Abschnitte herausgreifen möchte, dann ist die Cassette zu langsam, weil man immer erst die entsprechende Bandstelle suchen muß. Diese Anwendung fällt etwa an, wenn man aus einer Lagerliste mit Hunderten von Artikelnummern ein ganz bestimmtes Teil mit Preis usw. heraussuchen will. Anstelle des Cassettenrecorders nimmt man hier ein Floppy-Disk-Laufwerk. Es handelt sich dabei um eine Art Plattenspieler, der auf einer flexiblen Magnetfolie mit Hilfe eines automatisch positionierten Schreib/Lesekopfes Informationen beliebig oft einschreibt und ausliest. Die runde Folie dreht sich dabei mit so hoher Geschwindigkeit, daß sie durch die Fliehkraft genügend steif wird. Jede Stelle der Platte kann innerhalb eines Bruchteils einer Sekunde angefahren werden. Insgesamt hat sie für etwa $^{1}/_{4}$ Million Zeichen Platz, die mit etwa 30 000 Zeichen pro Sekunde geschrieben oder gelesen werden. Die Kosten für Floppy-Disk-Laufwerke sind im Augenblick sehr stark in Bewegung. Als Anhaltspunkt können aber runde 2000 DM für die preiswertesten Ausführungen genommen werden. Man sollte jedoch unbedingt darauf achten, daß das Laufwerk an den eigenen Computer paßt. Bisher ist es nämlich leider nicht üblich, daß einheitliche Floppy-Disk-Schnittstellen benutzt werden.

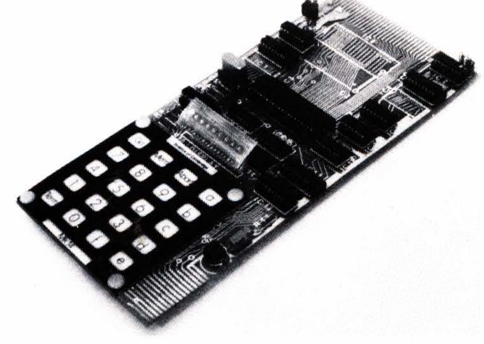
Die Preise von Druckern sind ebenfalls im Sinken. Momentan muß man für einen sogenannten Matrixdrucker, der jedes Zeichen aus einer Anordnung von 5 x 7 Punkten darstellt, noch etwa 2000 DM ausgeben. Damit kann man dann Großbuchstaben und Sonderzeichen darstellen.

Eine Zeile besteht üblicherweise aus etwa 80 Zeichen. Verschiedene Druckverfahren erfordern Spezialpapier, man kann also kein normales Schreibmaschinenpapier verwenden. Thermodrucker z. B. erhitzen die Punkte, die geschwärzt werden sollen, deshalb muß das Papier wärmeempfindlich sein. Metallpapierdrucker hingegen benötigen mit Aluminium beschichtetes Papier, aus dem sie die zu schwärzenden Punkte mit Stromfluß herausbrennen.

Prinzipiell können auch elektrische Schreibmaschinen mit Dateneingang verwendet werden. Allerdings erfordert die Anpassung an den Computer wieder schaltungstechnisches Wissen und ist deshalb für viele uninteressant. Außerdem sind derartige Schreibmaschinen sehr teuer. Mit der Anpassung alter



Der Computer ist in der Lage, Fragen zu stellen und die Antwort des Programmierers (in diesem Fall „Hund") in seiner Reaktion zu berücksichtigen. Auf diese Weise ist eine regelrechte Unterhaltung möglich. Die Handhabung von Texten und Wörtern nennen die Fachleute Stringverarbeitung



Durch seinen niedrigen Preis von knapp 200 DM fällt dieser Einkartencomputer mit dem Mikroprozessor SC/MP auf. Er hat, wie die meisten derartigen Systeme, bereits eine Tastatur und eine – allerdings winzige – Anzeigeeinheit auf der Leiterplatte. Um zu funktionieren, braucht er noch eine Stromversorgung

5-Kanal-Fernschreiber, die manchmal billig zu haben sind, verhält es sich genauso.

Generell kann man sagen: Beim Kauf eines Computers sind das Angebot und der Preis der Peripheriegeräte zu berücksichtigen. Erfahrungsgemäß kommt der Appetit beim Essen, und hat man erst einmal den Computer, dann stehen bald weitere Geräte auf der Wunschliste. Empfehlenswert für Nichttechniker ist der Kauf der für das eigene System vorgesehenen Zusatzeinheiten.

1.2 Computer-Grafik

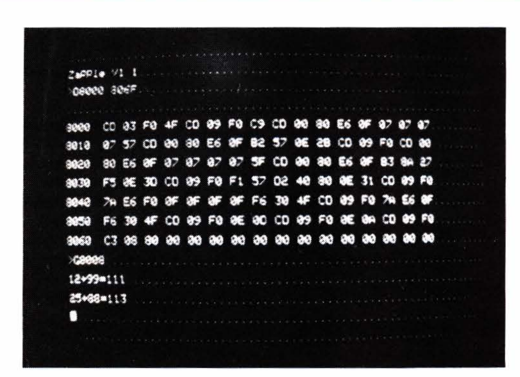
Ein großer Teil künftiger Computer-Hobbyisten wird sich aus heutigen Fans programmierbarer Taschenrechner zusammensetzen. Dieser Kreis kommt über die Darstellung von Funktionen (mit Bildschirm oder Drucker) und andere mathematische Problemstellungen sehr leicht in den interessanten Bereich der Computer-Grafik. Darunter ist weniger eine künstlerische Betätigung des Computers zu verstehen, sondern vielmehr generell die grafische Darstellung der vom Computer verarbeiteten Daten. Man benutzt dazu entweder den Bildschirm oder spezielle Ausgabegeräte (z. B. Plotter). Soll der Bildschirm benutzt werden, dann muß das System dafür ausgelegt sein. Normalerweise hat bei Heimcomputern nämlich die kleinste darstellbare Einheit die Größe eines Zeichens. Die Auflösung wird dadurch recht gering, und eine Kurve kann beispielsweise nur grob gestuft „gezeichnet" werden. Manche Systeme behelfen sich mit einem Trick: sie stellen verschiedene Grafiksymbole zur Verfügung, mit deren Hilfe man eine feinere Rasterung erzielt. Allerdings fehlt letztlich auch hier die entscheidende Möglichkeit, nämlich jeden Bildpunkt eines Bildschirms (z. B. 200 x 320) gesondert hell- oder dunkelsteuern zu können. Vorsicht ist geboten, wenn ein System diese Möglichkeit bietet, denn für die Abspeicherung dieser großen Anzahl von Bildpunkten muß natürlich ein entsprechend großer Speicher vorhanden sein. In der Grundausbaustufe ist dies wohl nie der Fall. Überschlägig kann man für eine „hochauflösende Grafik" 8 KByte an benötigtem Speicher (plus einige KByte Programmspeicher) ansetzen (was immer 8 KByte im Augenblick sein mögen), wofür man etwa 600 DM ausgeben muß. Auf dem Markt sind auch Computer, die über ein Farbfernsehgerät ein Bild in maximal 16 verschiedenen Farben ausgeben.

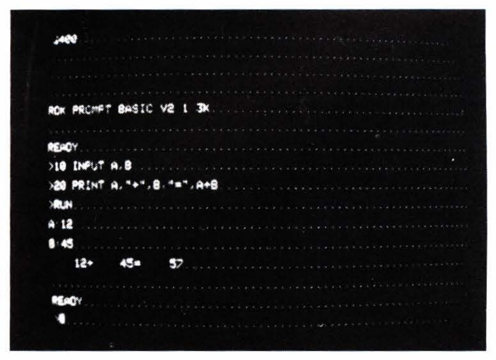
Eine andere Ausgabemöglichkeit für Computer-Grafik bieten Plotter. Das sind Zeichengeräte, bei denen man jeden Punkt der zur Verfügung stehenden Zeichenfläche individuell adressieren kann. Im Augenblick ist das Angebot noch gering. Die Schallmauer von 2000 DM wurde aber von einem Gerät bereits durchbrochen. Erfahrungsgemäß werden andere bald folgen.

1.3 Spiele

Die meisten Computer-Hobbyisten beginnen zunächst mit dem Programmieren von Spielen. Absicht ist es in den seltensten Fällen, diese Spiele auch wirklich zu benutzen. Die Übung im Programmieren und das Erfolgserlebnis, wenn das Programm läuft, sind fast immer der eigentliche Sinn. Ausnahmen sind Spiele, die auf einer sehr hohen Schwierigkeitsstufe stehen – etwa Schach.

Natürlich ist ein Schachprogramm eine sehr anspruchsvolle Aufgabe, die man nicht nach zwei Tagen in Angriff nehmen kann. Mancher wird dazu nie in der Lage sein, das sollte man nicht verschweigen. Trotzdem braucht man nicht auf solche Dinge zu verzichten. Das Computer-Hobby ist nämlich nicht – wie mancher vielleicht glaubt – eine Beschäftigung, die sich auf das stille Kämmerlein beschränkt. Im Gegenteil: es lebt von der Kommunikation mit Gleichgesinnten und von den Informationen, die man sammelt und in eigene Programme umsetzt. Bezogen auf Schach bedeutet das: Es gibt nur wenige grundlegend verschiedene Algorithmen (Ablaufvorschriften). Einige kluge Köpfe haben sich damit auseinandergesetzt. Aber Tausende von Programmierern haben diese Ideen aufgegriffen, verfeinert, mit zusätzlichen Kniffen ausgefeilt und schließlich ein eigenes Programm daraus gemacht. Die Aufgabe heißt also oft: geeignete





Der Programmieraufwand ist in Maschinensprache wesentlich höher als in BASIC: Das linke Bild zeigt ein Addierprogramm für zwei zweistellige Zahlen in Maschinensprache. Dazu werden 99 Byte an Befehlen benötigt. Die Ausführung wird mit „G8008" gestartet. Rechts das entsprechende BASIC-Programm (Zeilen 10 und 20)

Unterlagen über das Problem beschaffen – gute fremde Ideen mit eigenen verbinden – eigenes Programm daraus zu entwickeln.

2 Einkartencomputer

Eine völlig andere Anwendung als die Komplettsysteme haben die Einkartencomputer. Man will mit ihnen im allgemeinen irgendwelche Dinge steuern, die einen hohen Aufwand an Logik herkömmlicher Art (Gatter usw.) erfordern würden. Will man sich als Bastler damit beschäftigen, dann sollte man sich über zwei Dinge vorher im klaren sein:

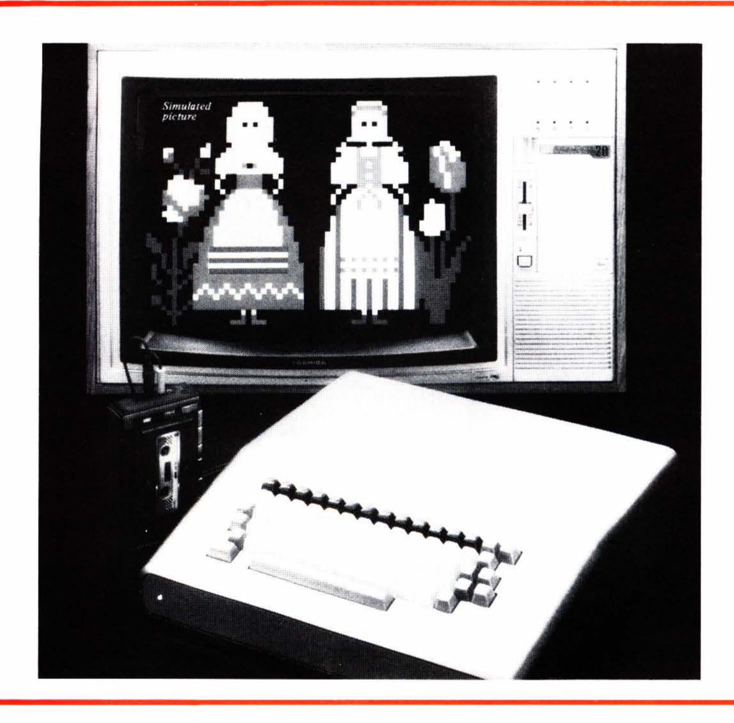
1. Der Anschluß externer Schaltungen, den man normalerweise anstrebt, verlangt ein Mindestmaß an schaltungstechnischen Kenntnissen und Möglichkeiten. Bastlern, die bisher etwa ausgiebige Erfahrungen mit der Digitaltechnik gesammelt haben, dürften keine unüberwindlichen Probleme erwachsen; vollkommenen Laien ist aber davon abzuraten.

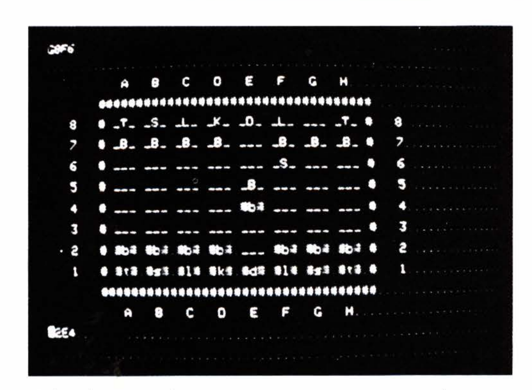
2. Sollte man Datenverarbeitung im Sinne von Rechnen oder Textverarbeitung anstreben, ist ein Einkartencomputer die falsche Ausstattung. Ausnahmen bilden Systeme, die man von einer Grundausstattung an mit Hilfe von Einschubkarten nahezu beliebig ausbauen kann (z. B. S-100-Bus).

Geht man von einem System aus, das mit Sedezimal-Tastatur und 7-Segment-Anzeigeeinheiten ausgestattet ist (z. B. KIM-1, SDK-85-Kit) und für ca. 800 DM angeboten wird, dann kann man damit z. B. folgende Dinge realisieren:

- eine Uhr, die den Langwellensender DCF 77 (Mainflingen) als Zeitnormal benutzt und demzufolge eine Abweichung von theoretisch 10^{-13} hat;
- eine Sprechererkennungseinheit, die aufgrund der Frequenzaufteilung der Stimme eine Person identifiziert und anzeigt;
- einen Musikerzeuger, der Töne in zufälliger oder gewünschter Reihenfolge spielt;
- einen Morsecode-Konverter, der bestimmte Normtexte auf Tastendruck aussendet (prinzipiell ist auch zeichenweise Aussendung und Decodierung von Texten möglich, die Schwierigkeit besteht aber in der beschränkten Zahl von Tasten und der schlechten Darstellbarkeit von Buchstaben mit 7-Segment-Anzeigeeinheiten – Abhilfe schafft ein Datensichtgerät);
- eine Datenerfassungseinheit, die Analogwerte wie Spannung, Strom, Temperatur, Frequenz, Lichtstärke usw. registriert, auf Grenzwertüberschreitungen überprüft und die Ergebnisse mit Kommentar über einen Billigdrucker ausgibt. Für eine solche Anwendung sind, im Gegensatz zu den unter Punkt 1.1 erwähnten Druckern, Typen mit etwa 20 Zeichen/Zeile geeignet, die man schon für 300...400 DM bekommt. Sie arbeiten meist mit Metall- oder Thermopapier;
- einen Schreibautomaten (mit Hilfe einer elektrischen Schreibmaschine), mit dem man Briefe entwerfen, verbessern und in beliebiger Zahl ausdrukken lassen kann.

Alle diese Anwendungen benötigen zusätzliche Schaltungen, die mehr (Langwellenempfänger) oder weniger (Lautsprecher) aufwendig sein können. Natürlich kann man denselben Mikrocomputer für verschiedene Aufgaben benutzen, gerade das ist ja seine Stärke. Programmiert wird in Maschinensprache, d. h. man muß sich mit dem Innenleben des Mikroprozessors auseinandersetzen. Ganz problemlos geht das nicht. Wenn man aber weiß, was ein Flußdiagramm ist und das Dualsystem kennt, dann kommt man bei einiger Übung schnell vorwärts. Gespeichert werden die Programme auf Musikcassetten oder in Festwertspeichern (ROMs).





Schach ist auch für einen Computer ein schweres Spiel. Hier ein Beispiel für die Darstellung von Feld und Figuren nach den ersten drei Zügen

◀ Eines der wenigen Geräte, die bereits in Farbe arbeiten: Es kommt von der japanischen Firma Toshiba, wird in BASIC programmiert und ist an einen normalen Farbfernseher anschließbar. In Deutschland wird es im Frühjahr 1979 für knapp 2500 DM erhältlich sein. Das wesentlich bekanntere Vorbild „Apple" aus USA ist ab etwa 4500 DM zu haben, leistet aber auch mehr

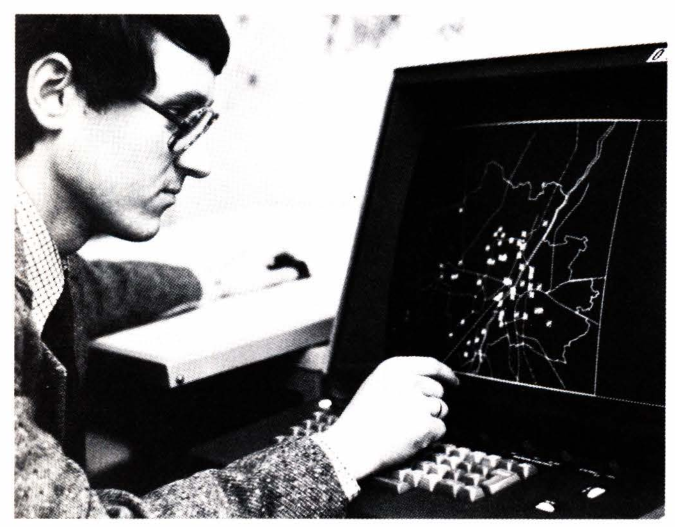
3 S-100-Bus

Werden Einkartencomputer um weitere Karten mit festgespeicherten Programmen (z. B. Übersetzer für BASIC) und zusätzlichen Funktionen sowie um Peripheriegeräte wie Datensichtgerät und Floppy Disk erweitert, dann entsteht natürlich ebenfalls ein volltauglicher Computer, der keinen Beschränkungen unterworfen ist. Das bekannteste System dieser Art ist unter dem Namen S-100-Bus bekannt. Es handelt sich dabei um eine Schnittstellen-Festlegung, die den Zusammenschluß von Karten verschiedener Hersteller gestattet. Durch die große Verbreitung des S-100-Bus wurden die Preise soweit gedrückt und werden soviele verschiedenartige Zusätze angeboten, daß für den Hobbyisten, der sein System ständig erweitern will, praktisch nichts anderes in Frage kommt. Man fängt üblicherweise mit einem Chassis an, das die Stromversorgung und mehrere Steckplätze für die unterschiedlichen Karten enthält. An jedem dieser Steckplätze kann nun eine beliebige S-100-Karte untergebracht werden – als erstes benötigt man natürlich einen Einkartencomputer. Eine weitere unentbehrliche Einheit ist das Datensichtgerät, d. h. die Steuerung, die dafür sorgt, daß man mit Hilfe von Tastatur und Bildschirm (unter Umständen normales Fernsehgerät) Daten darstellen und in den Mikrocomputer eingeben kann. Speicher sind ebenfalls erforderlich, und zwar RAMs und ROMs. RAMs sind Speicher, in die man beliebig oft einschreiben kann. ROMs hingegen werden einmal „gebrannt" und verlieren dann ihre Information nicht mehr, auch nicht, wenn die Stromversorgung abgeschaltet wird. Man verwendet sie deshalb für Programme, die nicht geändert werden. Auf den Einkartencomputern beispielsweise ist das sogenannte Monitorprogramm, das für eine gewisse minimale Intelligenz des Systems sorgt, in ROMs untergebracht. Auch das Programm, das dafür sorgt, daß der Computer die Sprache BASIC versteht, kann in einem ROM stehen; es kann aber auch von Cassette oder Floppy Disk nach dem Einschalten jedesmal in den RAM-Bereich geladen werden. Eine Untergruppe der ROMs sind die PROMs (programmierbare ROMs), die im Unterschied zu den ROMs nicht in der Fabrik, sondern beim Anwender programmiert werden. Bestimmte PROMs verlieren ihre Information, wenn man sie mit ultraviolettem Licht bestrahlt, sie sind löschbar und können neu programmiert werden, man nennt sie EPROMs (erasable PROM). Die entsprechenden Programmier- und Löschvorrichtungen sind ebenfalls als S-100-Karte zu haben.

In spezielle Richtungen gehen Zusatzeinheiten etwa für die Spracherkennung oder Bildverarbeitung. Unter Spracherkennung versteht man die Identifizierung bestimmter Wörter (z. B. Zahlen). Sie läuft darauf hinaus, daß man Computer akustisch bedienen kann. Zusammen mit der Sprechererkennung wird das in Zukunft große Bedeutung erlangen (Überweisung per Telefon o. ä.).

Die Bildverarbeitung ist eines der interessantesten Gebiete auf dem Computersektor überhaupt. Daß sie den Hobbyisten nicht verschlossen ist, haben drei junge „Forscher" beim Wettbewerb „Jugend forscht" gezeigt, die mit Hilfe einer beinahe primitiven Vorrichtung Bilder von Sternen digitalisiert und in einem Mikrocomputer weiterverarbeitet haben. Indem sie die Ergebnisse vieler Einzelbilder überlagerten, erhielten sie wesentlich detailliertere Aufschlüsse der Sternoberfläche, als man sie bisher hatte.

Noch ein Wort zum Preis von S-100-Bus-Systemen: Für ein Grundsystem (mit Sichtgerät und BASIC) muß man mit 4000 DM rechnen; Ergänzungskarten sind je nach Funktion für 200...1500 DM erhältlich.





Die Darstellung auf dem linken Bildschirm wäre mit dem PET (rechts) nicht möglich. Er kann nicht jeden Bildpunkt gesondert ansprechen. Allerdings stellt er eine Anzahl von Grafiksymbolen zur Verfügung, mit denen man ebenfalls recht phantasievolle Bilder erzeugen kann – z. B. das Raumschiff Enterprise, ein beliebtes Spielobjekt für Hobby-Programmierer

4 Ausblick

Im letztgenannten Beispiel verbindet sich das Computer-Hobby mit anderen Interessen in idealer Weise, und der Computer wird wieder zu dem, was er eigentlich sein sollte – zu einem sehr leistungsfähigen Werkzeug. Ähnliche Möglichkeiten stehen auch den Funkamateuren im Zusammenhang mit Funkfernschreiben und Schmalbandfernsehen offen.

Aus dem täglichen Leben werden Kleincomputersysteme bald nicht mehr wegzudenken sein. Geräte nach der Art des PET stehen im Augenblick gerade am Anfang und werden von Leuten gekauft, die Interesse am Programmieren haben. Diese Amateure werden aber in kurzer Zeit soviel Programme schreiben und auch zum Verkauf anbieten, daß eine solche Anschaffung auch für die reinen Benutzer interessant wird. Friseurläden, Bäckereien und Landwirte werden dann in den Kleinanzeigen der Tageszeitungen Programmangebote für die Abrechnung der Trinkgelder oder für die nächste Einkommensteuererklärung finden. Die Mikrodatentechnik wird nach der Groß-EDV und der mittleren Datentechnik ein weiterer bedeutender Entwicklungsschritt in Richtung Datenverarbeitungs-Zukunft sein.

Fachleute rechnen damit, daß in zwei bis vier Jahren Geräte auf den Markt kommen werden, die den heutigen Taschenrechnern sehr ähnlich sein werden. Die Miniaturisierung wird demnach auch weiterhin fortschreiten. Die Preise von Heimcomputern werden allerdings nicht mehr allzu stark fallen – eher wird man mehr Leistung hineinpacken, z. B. im Zusammenhang mit Bildschirmtext und Teletext. Der mögliche Zugriff auf Großdatenbanken, die etwa den gesamten Inhalt des großen Brockhaus enthalten könnten, wird für den Heimcomputer weniger eine Konkurrenz, sondern vielmehr ein weiteres Betätigungsfeld darstellen.

Angesichts solcher Zukunftsaussichten mag mancher von Ängsten befallen werden. Da aber Ängste oft aus der Unkenntnis entstehen, kann die Beschäftigung mit dem, was uns bevorsteht, nur von Vorteil sein.

Der Weg in ein faszinierendes Steckenpferd:

Tips für Anfänger im Computerhobby

Beobachter der Szene sind sich da sicher, obgleich sie ihre Meinung nicht auf präzise erhobenes Zahlenmaterial stützen können: Die Mehrzahl der Computer-Hobbyisten ist fachlich vorbelastet, sie bringt aus Lehre, Studium oder Berufsleben allerlei an Grundkenntnissen ins Steckenpferd ein. Die Pioniere der ersten Stunde sind oft genug von Anfang an ein bißchen Profi. Seit wenigen Monaten jedoch nimmt die Zahl der ‚reinrassigen Amateure' im Computerhobby-Bereich zu. Leute ohne Ausbildung oder Berufserfahrung in Technikbereichen entdecken ihre Liebe zur elektronischen Datenverarbeitung – viele von ihnen kommen aus dem Lager der Elektronik-Bastler und Freizeit-Funker, viele von ihnen haben allein Erfahrung mit analoger Signalverarbeitung. An diesen ‚Novizen-Kreis', dem er selbst entstammt, wendet sich der Autor dieses Beitrages mit einigen subjektiven Tips und Gedanken.

Zu Beginn eine Vorbemerkung: Wenn im folgenden einige Lehrgänge, Bausätze und Geräte genannt werden, so möge der Leser nicht daraus schließen, daß sie aus einer umfassenden Kenntnis des unterdessen doch recht reichlich beschickten Marktes heraus ausgewählt wurden: Dies ist nicht der Fall. Der Autor hält es für möglich, daß es über die erwähnten Erzeugnisse hinaus zum Zeitpunkt des Erscheinens dieses Beitrages sicherlich Konkurrenzprodukte gibt, die möglicherweise besser und billiger sind, als die, mit denen er selbst arbeitete. Daraus folgt, daß hier nur ein Weg von vielen möglichen beschrieben ist – nicht notwendigerweise der beste, schnellste und billigste, aber ein erprobter und, wie der Autor meint, ein ganz gut gangbarer.

Erste Erfahrungen in Digitaltechnik vermittelt da zum Beispiel der „Digitrainer" von ITT Fachlehrgänge, Pforzheim. Für rund 300 Mark bekommt man ein Experimentierfeld mit Netzteil, einigen Hilfseinrichtungen zur Erzeugung definierter digitaler Signale und zu ihrer Analyse: Pegelindikatoren mit Leuchtdioden, eine Siebensegment-LED. Dazu: eine Handvoll Gatter und MSI-Bauteile bis zum Dekadenzähler. Das mitgelieferte Lehrmaterial beschreibt, didaktisch gut gemacht, eine Fülle von Experimenten, die man durch das Einsetzen von ICs und das Stöpseln fertiger Verbindungsleitungen vornimmt. Wer den oft einige Konzentration erfordernden und bisweilen etwas anstrengenden, aber vorzüglich ‚lehrenden' Digitrainer-Lehrgang absolviert hat, ist für die höheren Sphären der Digital-Elektronik wohlgerüstet.

Billiger, wenngleich durchaus nicht schneller, ein anderer Einstieg in die Digital-Elektronik: Der Weg über die Konstruktion eines Zählers. Wer sich seinen Zähler aus elementaren Bauteilen wie dem 7490 zusammenbaut und dabei eine eigene Steuerung entwirft, auch ein wenig mit verschiedenen Lösungen herumspielt und dabei in Versuchsschaltungen das Verhalten von Flipflops der verschiedensten Art, von Gattern und Decodierern studiert – auch der ist fit für die Stufe zwei des Einstiegs ins Mikrocomputer-Geschäft – und die kann so aussehen:

Man kaufe eine ‚ALU', eine Recheneinheit-Schaltung wie etwa den 74181 von Texas Instruments. Hier kann man erstmals, an einer integrierten Schaltung für rund 7 Mark, komplexere digitale Datenverarbeitung erleben. Zwei 4-bit-Worte werden von dem Vierundzwanzigbeiner auf die unterschiedlichste Art miteinander verknüpft und zu einem 4-bit-Ausgang durchgeschaltet, und zwar abhängig von einen 5-bit-Steuerwort, das man an die Kommando-Eingänge der ALU anlegt. Die ALU beherrscht bereits alle elementaren logischen und mathematischen Verknüpfungen mit Ausnahme gewisser Verschiebe-Prozeduren, die auch zum Handwerkszeug der Mikroprozessoren gehören: Das ganze allerdings rein statisch, es gibt keinen Takt – das heißt, alles ist noch schön überschaubar und transparent.

Nächster Schritt: Jetzt besorge man sich ein 4-bit-Schieberegister mit paralleler Ein- und Ausgabe (z. B. 7495), probiere mal, was passiert, wenn man den ALU-Ausgang in eine Eingangsgruppe des Registers und Registerausgänge zurück in einen der beiden Datenkanäle der ALU führt, takte das Register am Shift- oder Load-Eingang (aber bitte mit Einzelimpuls, nicht durch ‚Antippen' eines Eingangs mit einer Masse-Strippe, weil dabei gleich ein ganzes Impulspaket auf die Reise geht). Und damit wäre man, wenn man versteht, warum das geschieht, was passiert, damit wäre man würdig für die höheren Weihen der Datenverarbeitung. Das heißt, in diesem Stadium wird es Zeit, sich dem Mikroprozessor selbst zuzuwenden.

Wer unter der 1000-Mark-Grenze bleiben muß, der kann sich mit dem Kit SDK 85 von Intel oder mit der berühmten Fertigkarte KIM den Grundstein für das zentrale Rechen- und Steuerwerk seines Hobbycomputer-Systems legen. Wer ein wenig mehr Geld ausgeben kann und schneller vorankommen möchte, dem kann aus Erfahrung der Mikroset 8080 von Siemens empfohlen werden oder ein anderes der Fix-und-fertig-Systeme, die in jüngster Zeit auf den Markt kamen.

Höhepunkt des Ausbaues eines Heimcomputer-Systems schließlich: BASIC-Maschinen von der Art des PET 2001, Bildschirm-Terminal und Telex- oder Teletype-Maschine und schließlich Mini-Floppies.

Dies also einige Hinweise, ein möglicher Weg zum Heimprogrammierer mit eigenem Computer. Man kann sicher sein, daß neue Erzeugnisse – Geräte, Lehrgänge, Fachbücher – den Computer-Novizen in nächster Zeit einiges an Nützlichem bieten werden, was ihm den Zugang zum faszinierenden Computerhobby etwas weniger dornenreich machen wird, als er es für die Anfänger der ersten Stunde war.

Hans-Georg Joepgen

Rudolf Hofer

Computer im Westentaschenformat

Ähnlich, wie beim Menschen das Gehirn, ist beim Computer die Zentraleinheit der wesentliche Teil. Durch die großartigen Leistungen der modernen Halbleitertechnologie ist es heute möglich, eine Zentraleinheit in einer einzigen integrierten Schaltung unterzubringen. Man nennt einen solchen Baustein Mikroprozessor (μP). Wie das Gehirn Arme, Beine, Augen, Ohren, Nase und Mund braucht, um Informationen aufnehmen und das Denkergebnis in die Tat umsetzen zu können, benötigt der Mikroprozessor Peripherieeinheiten, um funktionsfähig zu sein – durch diese Erweiterung wird er zum Mikrocomputer (μC). Da Mikrocomputer um Größenordnungen billiger und kleiner sind als herkömmliche Computer, sind sie im Augenblick dabei, in alle Bereiche unseres Lebens einzudringen: Sie sind in Haushaltsgeräten, in Autos, in Telefonvermittlungsanlagen, in Verkehrsampelsteuerungen und in Fernsehgeräten zu finden. Was man mit Mikroprozessoren und Mikrocomputern alles anfangen kann und wie sie programmiert werden, erfahren Sie auf den nächsten Seiten.



Wie kam es zum Mikroprozessor?

Um diese „Produkte der Zukunft" besser verstehen zu können, wollen wir zunächst einen Blick in die Vergangenheit werfen: Es ist noch gar nicht so lange her, da haben die drei Amerikaner John Bardeen, Walter Brattain und William Schockley den Transistor erfunden – vor dreißig Jahren war das. In diese Zeit fielen auch die Anfänge der Computertechnik. Man hatte damals Relaisrechner, bei denen alle Vorgänge mit mechanischen Schaltelementen ausgeführt wurden. Auch mit einem Rechner, der 18 000 Röhren enthielt, wurden Versuche angestellt. Natürlich war er praktisch eher zum Heizen als zum „Rechnen" zu gebrauchen, denn zumindest eine Röhre war immer kaputt. Die Erfindung des Transistors brachte die Entwicklung ein ganzes Stück weiter, obwohl nach heutigen Begriffen auch ein mit Einzeltransistoren aufgebauter Computer einen unvorstellbaren Aufwand bedeutet. Man stelle sich nur vor, daß zur Speicherung eines einzigen Zeichens (etwa eines Buchstabens) schon mindestens 12 Transistoren nötig waren. Es dauerte immerhin bis zum Jahr 1960, bis der erste Computer dieser Art vorgestellt wurde.

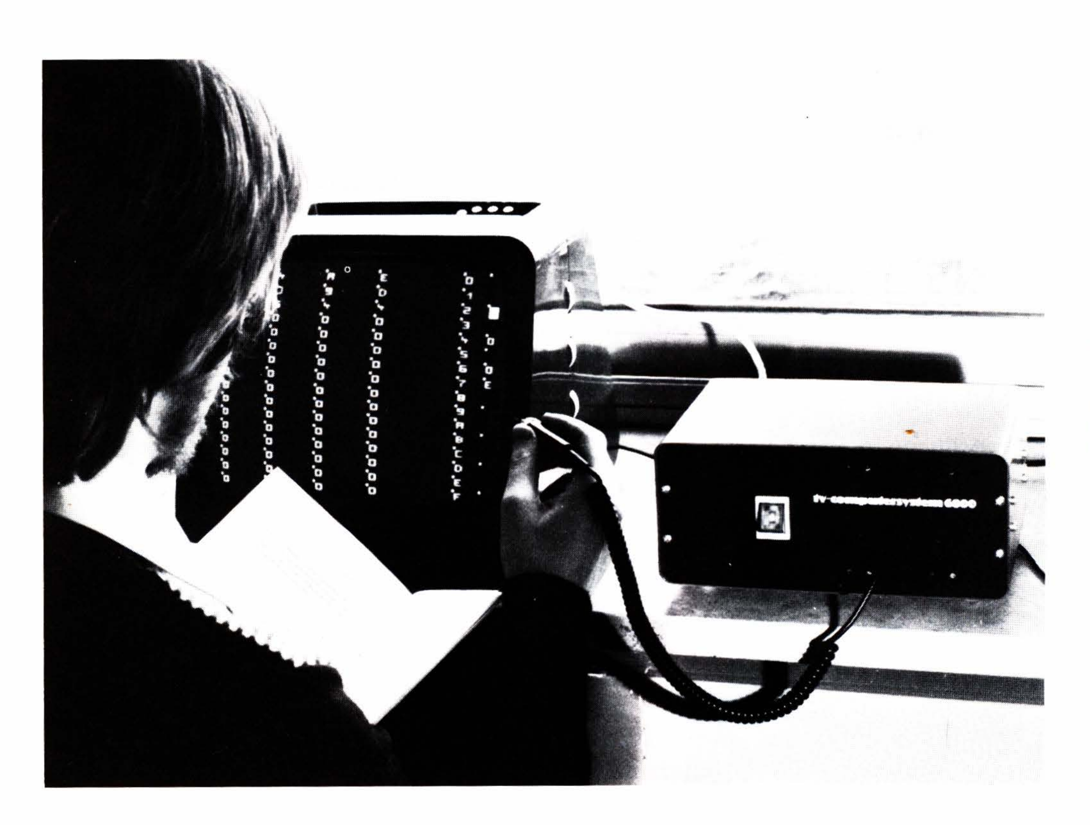
Und wieder folgte diesem Schritt in der Computertechnik ein bedeutungsvolles Ereignis in der Halbleitertechnik: Anfang der sechziger Jahre gelang es erstmals, mehrere Transistoren auf demselben Halbleiterstück zu „integrierten". Von da ab ging es sehr schnell. Man verstand es, immer mehr Transistoren auf immer kleinerem Raum unterzubringen. Hatte man zu Beginn nur einfache Gatter hergestellt, so wurden es im Laufe der Zeit Flipflops, Zählerstufen, Decodierer, Schieberegister und Addierer. Immer spezieller wurden die Funktionen. Aus Gattern kann man prinzipiell noch jede Schaltung verwirklichen, aber ein Addierer kann eben nur addieren. Am Ende dieser Entwicklung standen unter anderem Taschenrechner-, Orgel- oder Uhrenschaltungen, die in sehr großen Stückzahlen hergestellt wurden (und immer noch werden) – nur durch die große Anzahl können solche Schaltungen nämlich billig verkauft werden. Doch nicht jeder brauchte Taschenrechner-, Uhren- oder Orgelschaltungen: Beim weitaus größten Teil der Anwendungen wurden andere Funktionen verlangt, die im einzelnen nur in geringen Stückzahlen benötigt wurden. Für die Lösung dieser Probleme mußte man weiterhin auf relativ einfache integrierte Schaltungen zurückgreifen.

Um die technologischen Möglichkeiten der „Großintegration" allgemein nutzen zu können, suchte man nach einem Baustein, mit dem man jedes Problem lösen konnte – nach einem Baustein also, der universell verwendbar war und deshalb in großen Stückzahlen hergestellt werden konnte.

Anfang der siebziger Jahre sorgte die amerikanische Firma Intel für des Rätsels Lösung: Sie stellte den ersten Mikroprozessor vor. Und siehe da: es handelte sich tatsächlich um einen Computer im Kleinstformat.

Mikroprozessoren ersetzen meistens festverdrahtete Schaltungen

Freilich werden Mikroprozessoren im allgemeinen nicht für typische Computeraufgaben eingesetzt. Meistens lösen sie sogenannte festverdrahtete Logikschaltungen ab. Man versteht darunter Schaltungen, die für einen einzigen, ganz bestimmten Zweck, geeignet sind. Stellen wir uns beispielsweise eine einfache Aufzugsteuerung vor (die es in der Form sicher nicht gibt): Die festverdrahtete Logik sorgt dafür, daß der Aufzug die Stockwerke in derselben Reihenfolge an-



Dieser „TV-Computer" ist in erster Linie für die Ausbildung von „Profis" gedacht: Für die Dateneingabe wird ein Lichtgriffel verwendet

fährt, in der die entsprechenden Knöpfe betätigt worden sind. Nun fällt es dem Besitzer ein, daß der Aufzug ruhig etwas „intelligenter" sein könnte. Er soll jetzt – wenn er sich z. B. im vierten Stock befindet – noch eine Person aus dem fünften Stock mitnehmen, bevor er in den Keller fährt. Mit der bisherigen festverdrahteten Steuerung kann man diese Aufgabe nicht mehr lösen; es muß eine vollkommen neue Schaltung entworfen werden. Wäre die Aufzugsteuerung mit einem Mikroprozessor aufgebaut gewesen, dann hätte dieselbe Schaltung auch die neue Aufgabe ausgeführt – man hätte lediglich das Programm ändern müssen (wie man das macht, sehen wir später). Damit sind wir bereits bei der wesentlichen Eigenschaft des Mikroprozessors: Er ist programmierbar. Ähnlich wie beim menschlichen Gehirn das im Laufe der Zeit angesammelte Wissen darüber entscheidet, ob es später in der Lage ist, Fremdsprachen zu verstehen, mathematische Formeln zu berechnen oder Noten zu lesen, entscheidet beim Mikroprozessor (oder besser beim Mikrocomputer) das Programm darüber, ob er einen Aufzug oder eine Ampel steuern soll. Das Programm ist also die Anweisung, was der Computer zu tun hat, es wird vom Programmierer vorgegeben. Im Grunde muß also der Programmierer das gestellte Problem lösen, und er muß den Lösungsweg dem Computer (in verschlüsselter Form) mitteilen. Der Computer wird also niemals eine eigene Entscheidung treffen, sondern immer nur das nachvollziehen, was der Programmierer für ihn vorgedacht hat. Dieses Nachvollziehen tut er aber in der Regel wesentlich schneller, als es ein Mensch könnte. Auch Mikroprozessoren arbeiten für menschliche Begriffe sehr schnell, trotzdem sind sie (noch) zu langsam, um etwa heutigen Großcomputern Konkurrenz zu machen. Für die Anwendungen, die man als Hobby-Elektroniker ins Auge fassen könnte, reicht aber die Geschwindigkeit allemal aus.

Digitale Schaltungen kennen nur zwei Zustände

Elektronische Schaltungen teilen sich nach ihrer Arbeitsweise in zwei Arten auf: in analoge und digitale. Analoge Schaltungen verarbeiten stetige Signale, d. h. die Form des Signals ist von entscheidender Bedeutung. Der Verstärker eines Plattenspielers z. B. hat nichts anderes zu tun, als die vom Tonabnehmer gelieferte Spannung soweit zu erhöhen, daß ein Lautsprecher betrieben werden kann. Der Spannungsverlauf soll am Ausgang aber möglichst dieselbe Form haben wie am Eingang.

Digitale Schaltungen – zu denen auch Mikroprozessoren gehören – verarbeiten Signale, die nur eine ganz bestimmte Anzahl von Zuständen annehmen können. Im Normalfall sind das zwei Zustände (wir sprechen in diesem Fall von Binär-Schaltungen oder binärer Logik). Man ordnet ihnen die Zahlen „0" und „1" zu. Meistens bedeutet „0", daß die Spannung niedrig ist (z. B. 0 V), während „1" den hohen Spannungspegel (z. B. 3,5 V) angibt. Ein binäres Signal läßt sich deshalb auch als Folge von Einsen und Nullen darstellen, wie es *Bild 1* zeigt.

Die Tatsache, daß man nur zwei Zustände darstellen kann, bedeutet aber nicht, daß man in der Digitaltechnik auf zwei Zahlen beschränkt ist. Auch im Dezimalsystem, dem uns allen geläufigen Zahlensystem, kennt man nur eine beschränkte Anzahl von darstellbaren „Zuständen" – die zehn Ziffern 0...9. Trotzdem kann man mit diesen zehn Ziffern unendlich viele Zahlen darstellen. Hat man nur zwei Ziffern (0 und 1) zur Verfügung, dann kann man ebenfalls unendlich viele Zahlen darstellen – allerdings im Zweiersystem (auch Dualsystem oder Binärsystem genannt), das auf Seite 20 näher besprochen wird. Eine Stelle einer Dualzahl nennt man Bit*. Mit 1 bit lassen sich also 2^1 (= 2) Zustände darstellen, mit 2 bit 2^2 (= 4), mit 3 bit 2^3 (= 8) usw., demnach ergeben sich für 8 bit 2^8 (= 256) Möglichkeiten. In unserem Beispiel (*Bild 2*) entspricht jeder Schalter einem Bit (weil er in genau zwei Stellungen gebracht werden kann). Mit vier Schaltern können wir – nach dem, was wir bisher gelernt haben – 2^4 (= 16) verschiedene Ein/Aus-Kombinationen herstellen, das sind 16 verschiedene „Leuchtkombinationen". Fassen wir jetzt „Leuchten" (= Schalter ein) als „1" auf und „nicht Leuchten" (= Schalter aus) als „0" und legen dann noch fest, daß die niederwertige Stelle (2^0) rechts stehen soll, dann können wir mit

* Bit wird als Einheit meist klein geschrieben

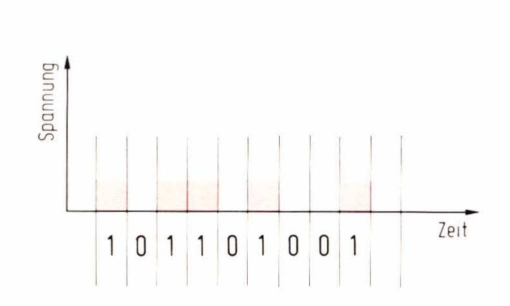


Bild 1. Alle Digitalschaltungen arbeiten heute binär, d. h. es gibt nur zwei erlaubte Zustände des Signals: „unten" und „oben" – „in der Mitte" hat es nichts zu suchen, außer beim Wechsel

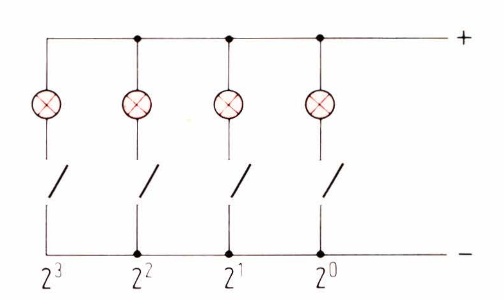
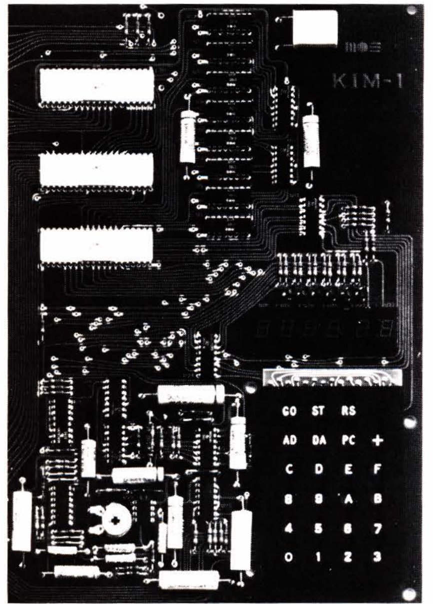
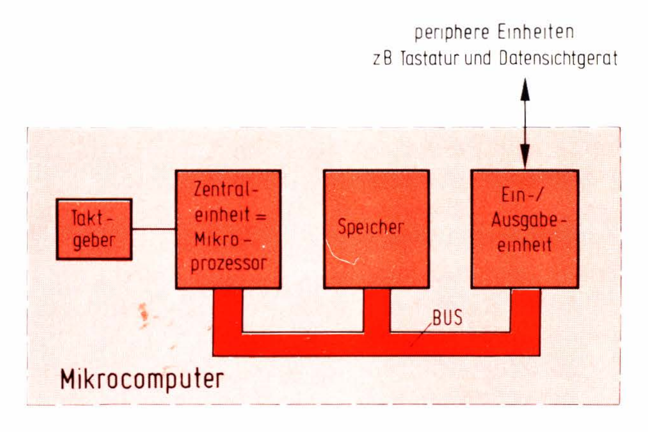
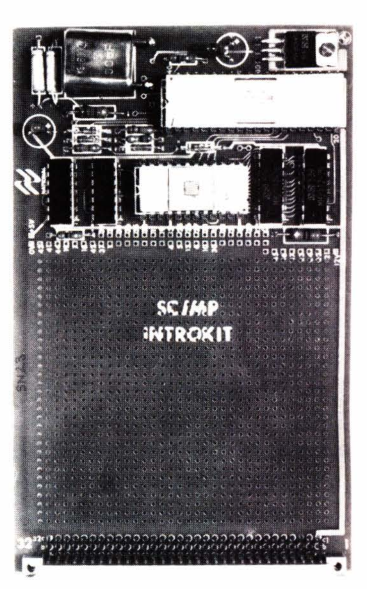
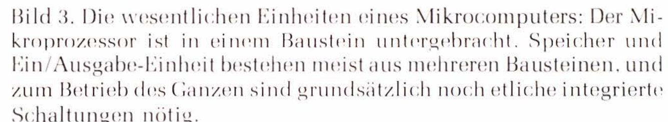


Bild 2. Mit vier Lämpchen kann man 16 Dualzahlen darstellen, mit sieben Lämpchen sind es schon 128. 7 bit reichen aus, um Buchstaben, Zahlen und Sonderzeichen vereinbaren zu können








Bild 3. Die wesentlichen Einheiten eines Mikrocomputers: Der Mikroprozessor ist in einem Baustein untergebracht. Speicher und Ein/Ausgabe-Einheit bestehen meist aus mehreren Bausteinen, und zum Betrieb des Ganzen sind grundsätzlich noch etliche integrierte Schaltungen nötig.
Der Aufwand für einen Mikrocomputer kann recht unterschiedlich sein, wie die Bilder rechts und links beweisen

den Lämpchen Dualzahlen darstellen. Einem Freund, dem wir dies vorher mitgeteilt haben, könnten wir auf diese Weise der Reihe nach Zahlen übermitteln.

Was Daten bedeuten, bestimmen wir

Nun hat es wenig Sinn, jemandem Zahlenreihen zu übermitteln. Interessanter wäre die Sache schon, wenn dasselbe auch mit Buchstaben ginge. Aber was hindert uns daran, zu vereinbaren, daß die Dualzahl 0001 ein A sein soll, die Dualzahl 0010 ein B, die Dualzahl 0011 ein C usw. Wir sind auch nicht auf Buchstaben begrenzt: Eine bestimmte Bit-Kombination könnte z. B. bedeuten „Schalte das Licht aus!", eine andere „Schalte das Radio ein!". Allgemein gesagt: Wir können Daten darstellen – denn Daten sind alle Dinge und Sachverhalte, die sich in binärer Form darstellen lassen. Es kommt nur darauf an, was wir uns darunter vorstellen wollen.

Ersetzen wir unseren Freund jetzt durch einen Mikrocomputer und die Lämpchen durch Leitungen, an die wir verschiedene Bit-Kombinationen in Form von Spannungen (z. B. 0 V = „0", 3,5 V = „1") anlegen, dann können wir dem Mikrocomputer Daten eingeben oder auch Befehle übermitteln. Hier müssen wir nun ganz klar unterscheiden: Die Befehle, die der Mikroprozessor verstehen soll, sind bereits vom Hersteller festgelegt – wir können sie nur benutzen, nicht aber festlegen. Die Daten, die wir hineinschicken und die wir (aufgrund der Befehle) wieder erhalten, können von uns beliebig gedeutet und verwendet werden (als Zahlen, Buchstaben, Steuersignale für Lampen oder Relais usw.). Im weiteren wollen wir deshalb zwischen Daten und Befehlen einen Unterschied machen, obwohl im Grunde auch die Befehle Daten sind.

Funktionseinheiten eines Mikrocomputers

In *Bild 3* sind die wesentlichen Einheiten eines Mikrocomputers dargestellt. Sie sind über einen sogenannten Bus (parallele Leitungen) miteinander verbunden, über den Daten, Adressen (auf deren Bedeutung wir noch zu sprechen kommen) und verschiedene Steuersignale übertragen werden. Im Speicher steht zunächst einmal das Programm, er wird aber auch zur Zwischenspeicherung von Daten benutzt. Die Zentraleinheit (das ist der Mikroprozessor) „verarbeitet" die Daten so, wie es das Programm vorschreibt.

Über die Ein/Ausgabe-Einheit wird der Datenaustausch mit der Außenwelt vorgenommen, sie sorgt z. B. dafür, daß die parallel vorliegenden Daten in ein serielles Signal umgewandelt werden, damit man sie der Reihe nach auf einer einzigen Leitung übertragen kann. Umgekehrt wandelt sie ein von außen kommendes serielles Signal in parallele Daten um.

Der gesamte Datenfluß und die Verarbeitung laufen in einem bestimmten Rhythmus ab, der vom Taktgeber vorgegeben wird. Deshalb weiß man genau, wie lange es dauert, bis ein Befehl ausgeführt ist (meist einige Millionstel Sekunden [μs]). Der Taktgeber ist oft schon im Mikroprozessor selbst untergebracht.

Die Steuerleitungen haben beispielsweise den Sinn, der Ein/Ausgabe-Einheit die Richtung mitzuteilen, in der die Daten fließen sollen. Sie sind zwar für das Zusammenwirken der einzelnen Einheiten sehr wichtig, der Programmierer muß sich um ihre Funktion aber nicht kümmern, da die Steuersignale aufgrund des Programms automatisch erzeugt werden. Für die Steuersignale stehen immer getrennte Leitungen zur Verfügung. Bei Daten- und Adreßbus ist das anders: Sie können aus gemeinsamen Leitungen bestehen, in denen – zeitlich nacheinander – Daten und Adressen übertragen werden. Meistens stehen aber auch hier getrennte Leitungen zur Verfügung, und zwar 8 für den Datenbus und 16 für den Adreßbus. In diesem Fall sprechen wir von 8-bit-Mikroprozessoren, zu denen die meisten heute hergestellten Typen gehören. Sie verarbeiten 8 bit lange Daten, die wir auch als Bytes bezeichnen. Die Anzahl der Adreßleitungen hat damit nichts zu tun: Sie bestimmt die Zahl der Speicherplät-

ze, die man direkt ansprechen kann (bei einem 16-bit-Adreßbus sind das 65 536).

Speicher

Um die Funktion des Speichers verstehen zu lernen, stellen wir uns vor, wir hätten einen Taschenrechner zur Hand. Auf einem Blatt Papier (*Bild 4*), dessen Zeilen von oben nach unten durchnumeriert sind, steht die Anweisung für eine Berechnung. Wir wissen, daß wir bei Zeile 0 beginnen und dann solange den nächsten Befehl ausführen sollen, bis das „Programm" etwas anderes vorschreibt. Genauso arbeitet ein Mikroprozessor mit dem Speicher zusammen: Anstatt in verschiedene Zeilen ist dieser aber in Adressen aufgeteilt, die ebenfalls von 0 an durchnumeriert sind. Unter jeder Adresse kann 1 Byte gespeichert werden (also 8 bit). Bestimmte Bit-Kombinationen faßt der Mikroprozessor als Befehle auf (z. B als Additionsbefehl), die – wie wir bereits wissen – vom Hersteller festgelegt sind. Der Ablauf eines Programms ist also folgender: Zu Beginn sendet der Mikroprozessor die Anfangsadresse über den Adreßbus an den Speicher. Über den Datenbus schickt daraufhin der Speicher den unter dieser Adresse gespeicherten Befehl an den Mikroprozessor. Dieser führt den Befehl aus und stellt die Adresse fest, unter der der nächste Befehl zu finden ist usw. Soll der Mikroprozessor nur für eine Aufgabe verwendt werden, etwa zur Steuerung eines Aufzugs, dann bleibt das Arbeitsprogramm immer dasselbe, und man kann einen sogenannten Festwertspeicher (ROM) verwenden. Sein Inhalt wird einmal „eingebrannt" und kann dann nicht mehr verändert werden. Deshalb wäre ein ROM für die Speicherplätze 6 bis 8 in unserem Beispiel nicht geeignet. Wenn wir auch andere Zahlen als 4 und 2 addieren wollen, müssen wir diese Speicherplätze ja verändern können. Hier brauchen wir einen Schreib/Lese-Speicher (RAM), dessen Inhalt während des Programmablaufes beliebig geändert werden kann. Sein Nachteil ist, daß er die Information verliert, wenn die Versorgungsspannung abgeschaltet wird. Der Mikroprozessor macht keinen Unterschied zwischen ROM und RAM, deshalb kann man das Arbeitsprogramm genausogut auch in einem RAM speichern. Sinnvoll ist das, wenn man verschiedene Arbeitsprogramme verwenden will, und das wird bei den meisten Hobbyanwendungen der Fall sein. Außerdem braucht man besondere Einrichtungen zum Programmieren.

Das Innere des Mikroprozessors besteht aus Registern

Die immer schwieriger werdende Technik hat in der Elektronik dazu geführt, daß man nicht mehr versucht, die Funktion des einzelnen Elementes einer Schaltung bis ins Detail zu begreifen. Man faßt vielmehr ganze Schaltungsteile zu Funktionsblöcken zusammen, von denen man weiß, wie sie auf bestimmte Signale reagieren. Warum sie das tun, interessiert höchstens den Hersteller. Wir wollen deshalb die Arbeitsweise des Mikroprozessors ebenfalls nur unter dem Gesichtspunkt betrachten, was er mit den Daten macht, die wir ihm eingeben. Wie das im einzelnen funktioniert, ist für den Anwender völlig unwichtig. Für den Programmierer besteht das Innere des Mikroprozessors aus Registern. Register sind nichts anders als Speicher, in denen zu verarbeitende Daten und Adressen zwischengespeichert werden. Da in den meisten Fällen gleichzeitig jeweils 8 bit verarbeitet werden und die Adressen eine „Länge" von 16 bit haben, enthalten die meisten Mikroprozessoren hauptsächlich 8-bit-Register und wenigstens ein 16-bit-Register.

Eine wichtige Stellung nimmt der Akkumulator ein – das Arbeitsregister. Er kann 8-bit-Dualzahlen addieren, logische Verknüpfungen durchführen (was man sonst mit Gattern macht), Daten in Speicherzellen schreiben, aus dem Speicher holen oder an die Ausgabeeinheit weitergeben, Daten miteinander vergleichen usw. – Auf den ersten Blick wirken diese Möglichkeiten recht primitiv. Sie sind aber die Grundbausteine für komplizierteste Programme, auch für große Computer-Anlagen. Es wäre, nur um ein Beispiel zu nennen, durchaus möglich, die Funktionen eines programmierbaren wissenschaftlichen Taschenrechners aus diesen Grundoperationen nachzubilden. Allerdings, und das muß hinzugefügt werden, dürfte das selbst für einen „Profi" eine recht anspruchsvolle Aufgabe sein.

Programmzähler, Bedingungsregister und Stapelspeicher

Wenn wir uns das „Programm" von Bild 4 noch einmal ansehen, fällt uns auf, daß die Befehle in der Reihenfolge abgearbeitet werden, wie sie im Speicher stehen. Ein wesentliches Merkmal einer Datenverarbeitungsanlage ist aber gerade, daß dies nicht der Fall sein muß. Wie wir später noch sehen werden, gibt es Befehle, die den Rechner dazu veranlassen, an einer beliebigen Stelle im Programm fortzufahren. Das sind z. B. Sprungbefehle. Eine andere, ganz wichtige Eigenschaft von Computern ist, daß solche Programmsprünge von einer Bedingung abhängig gemacht werden können. Zur Darstellung eines Programmablaufs verwendet man sogenannte Flußdiagramme. *Bild 5* zeigt an einem praktischen Beispiel, wie sie ein Problem veranschaulichen. Im Grunde enthalten sie sogar schon die Lösung, deshalb ist ein Flußdiagramm der erste Schritt zur Erstellung eines Computerprogramms.

Im Inneren des Mikroprozessors (*Bild 6*) sorgen vor allem der Programmzähler und das Bedingungsregister dafür, daß solche Verzweigungen realisiert werden können. Im Programmzähler (ein 16-bit-Register) steht jeweils die Adresse des Befehls, der als nächster ausgeführt werden soll. Am Programmanfang wird er mit dem Inhalt von zwei festgelegten Speicherzellen geladen, unter denen der Programmierer folglich die Startadresse ablegen muß. Nachdem ein Befehl ausgeführt ist, wird der Inhalt des Programmzählers auto-

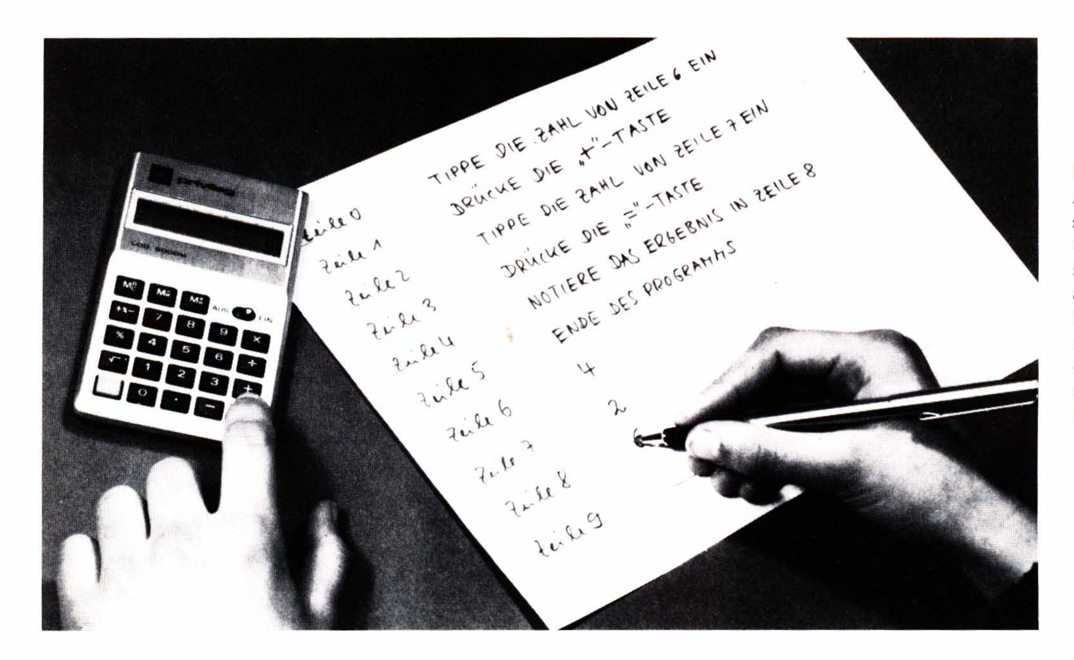
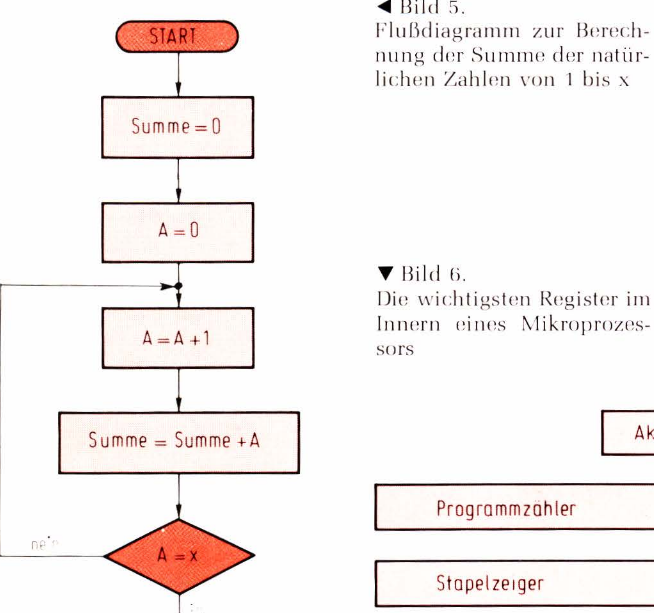


Bild 4. Der Zettel enthält die Anweisung, was in den Taschenrechner einzutippen ist und wo das Ergebnis hingeschrieben werden soll – er dient als Speicher. Der Speicher eines Mikrocomputers enthält die Anweisung (Programm), was der Mikroprozessor tun soll – selbstverständlich in binärer Form

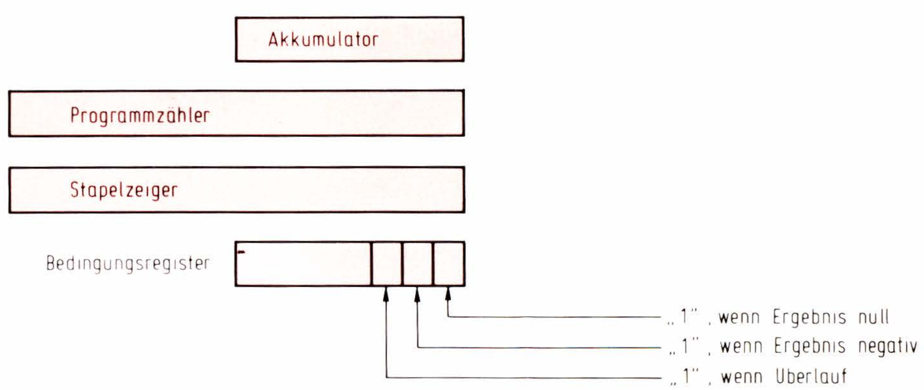
matisch erhöht. Sprungbefehle setzen den Programmzähler auf eine beliebige Adresse (unter dieser Adresse wird dann der nächste Befehl gesucht). In unserem Beispiel muß dem Vergleich A = x also ein Sprungbefehl folgen, der den Programmzähler auf die erste Adresse des Befehls A = A + 1 setzt (selbstverständlich kann der Inhalt eines Kästchens in einem Flußdiagramm aus mehr als einem Befehl bestehen). Allerdings muß eine Bedingung erfüllt sein, damit der Sprung ausgeführt wird (wir sprechen von einem bedingten Sprungbefehl), nämlich A ≠ x (A ungleich x). Ist diese Bedingung nicht erfüllt (d. h. A = x), wird der Programmzähler wie gewöhnlich erhöht, und der nächste Befehl („Summe ausgeben") wird bearbeitet. Die Frage, ob eine Bedingung erfüllt ist oder nicht, beantwortet das Bedingungsregister. Nach bestimmten (nicht allen) Operationen, wie „Addition", die man mit Hilfe des Akkumulators durchführt, werden einzelne Bits in diesem Register verändert. In unserem Fall wird z. B. Bit 0 auf „1" gesetzt, wenn das Ergebnis einer Operation null ist, Bit 1 wird gesetzt, wenn das Ergebnis negativ ist, und Bit 2 zeigt an, ob ein Überlauf auftritt. Letzteres ist beispielsweise der Fall, wenn zwei 8-bit-Zahlen addiert werden und das Ergebnis mehr als 8 Stellen hat.

Bei bedingten Sprungbefehlen wird nun der Zustand eines bestimmten Bits im Bedingungsregister (automatisch) abgefragt. Ist das Bit gesetzt, dann wird der Sprung ausgeführt. Für den Programmierer ist es wichtig, daß er genau weiß, wie das Bedingungsregister von den verschiedenen Befehlen beeinflußt wird.

Der Stapelzeiger ist das letzte Register von Bild 6, das noch nicht erklärt ist; er enthält ebenfalls eine Adresse. Um seine Funktion zu verstehen, muß man zunächst einmal wissen, welche Aufgabe der Stapelspeicher hat: *Bild 7* zeigt einen Programmablauf, wie er häufig vorkommt. Dabei muß an mehreren Stellen im Hauptprogramm dieselbe Aufgabe ausgeführt werden (hier die Berechnung der Wurzel). Die Be-



◀ Bild 5. Flußdiagramm zur Berechnung der Summe der natürlichen Zahlen von 1 bis x



▼ Bild 6. Die wichtigsten Register im Innern eines Mikroprozessors

fehlsfolge für diese Berechnung wird aber nur ein einziges Mal „niedergeschrieben“. Aus dem Hauptprogramm springt man dann dieses „Unterprogramm“ beliebig oft an, und zwar mit einem speziellen Sprungbefehl, der die Anfangsadresse des Unterprogramms enthält. Schwieriger ist der Rücksprung, da jetzt verschiedene Sprungziele vorhanden sind. Vor einem Unterprogrammaufruf muß deshalb die Rücksprungadresse irgendwo hinterlegt werden – der Mikroprozessor benutzt dazu den Stapelspeicher.

Nun ist es möglich, aus einem Unterprogramm (UP) in ein weiteres zu springen usw. (*Bild 8*). Die neue Rücksprungadresse wird jeweils wieder im Stapelspeicher aufbewahrt. Bei jedem Rücksprung muß dann die zuletzt eingegebene Adresse *zuerst* wieder ausgegeben werden – wie bei einem Kartenstapel, von dem man immer nur die oberste Karte wieder wegnehmen kann.

Der Stapelspeicher kann bei manchen Prozessoren bei einer beliebigen Adresse beginnen. Der Programmierer muß diese Adresse in den Stapelzeiger schreiben. Dieser weist in der Tat wie ein Zeiger auf eine Speicherzelle hin.

Beim Füllen des Stapelspeichers wird er – meist automatisch – erniedrigt, beim Auslesen wird er erhöht. Andere Prozessoren haben eine feste Anzahl von Registern als Stapelspeicher, prinzipiell arbeiten sie jedoch genauso.

Eine ähnliche Aufgabe hat der Stapelspeicher bei einer Programmunterbrechung (Interrupt), die zu einem unvorhergesehenen Zeitpunkt von außen ausgelöst werden kann. Der Prozessor muß in einem solchen Fall das Hauptprogramm sofort unterbrechen und ein anderes bearbeiten, erst nach Beendigung des „Unterbrechungsprogramms“ kann er im Hauptprogramm fortfahren. Damit er unter denselben Bedingungen weitermachen kann, wie sie vor dem Interrupt geherrscht haben, legt er alle Registerinhalte im Stapelspeicher ab und holt sie anschließend wieder zurück. Ein Interrupt könnte z. B. bei einer Ampelanlage ausgelöst werden, wenn eine Signallampe ausfällt. Das Unterbrechungsprogramm würde dann ein blinkendes Gelblicht zur Folge haben.

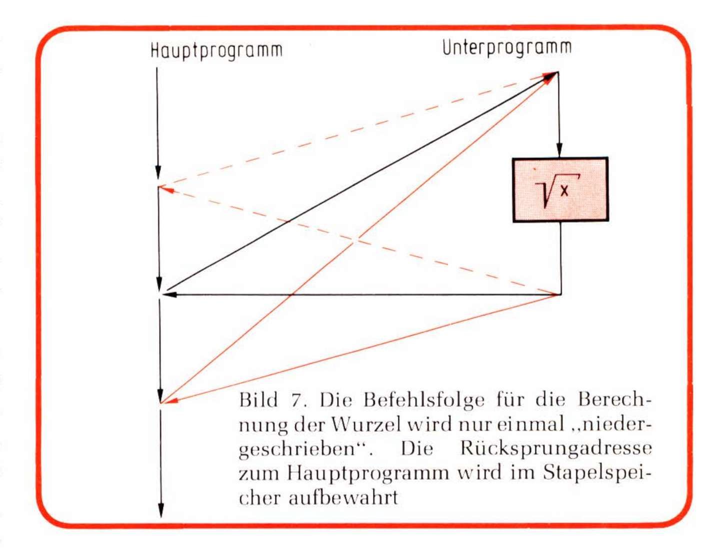


Bild 7. Die Befehlsfolge für die Berechnung der Wurzel wird nur einmal „niedergeschrieben“. Die Rücksprungadresse zum Hauptprogramm wird im Stapelspeicher aufbewahrt

Eine dritte Aufgabe des Stapelspeichers ist die Zwischenspeicherung von Daten.

Befehlsarten

Die meisten Mikroprozessor-Modelle „verstehen“ zwischen 50 und 200 Befehle. Die wichtigsten sind:

● *Transportbefehle:* Mit ihnen bringt man Daten vom Speicher in die Register und umgekehrt. Man braucht sie z. B., um Ergebnisse abzuspeichern oder Operanden in den Akkumulator zu holen. Ein/Ausgabe-Befehle (E/A) fallen ebenfalls unter diese Kategorie. Manche Prozessoren benötigen keine speziellen E/A-Befehle mehr, sondern betrachten Ein- und Ausgabe-Kanal als Teil des Speichers, der unter bestimmten Adressen zu erreichen ist.

● *Schiebebefehle:* Mit ihnen kann man den Inhalt des Akkumulators oder auch einer Speicherzelle um eine Stelle nach links oder rechts verschieben. Das Übertragsbit des Bedingungsregisters kann an dieser Ope-

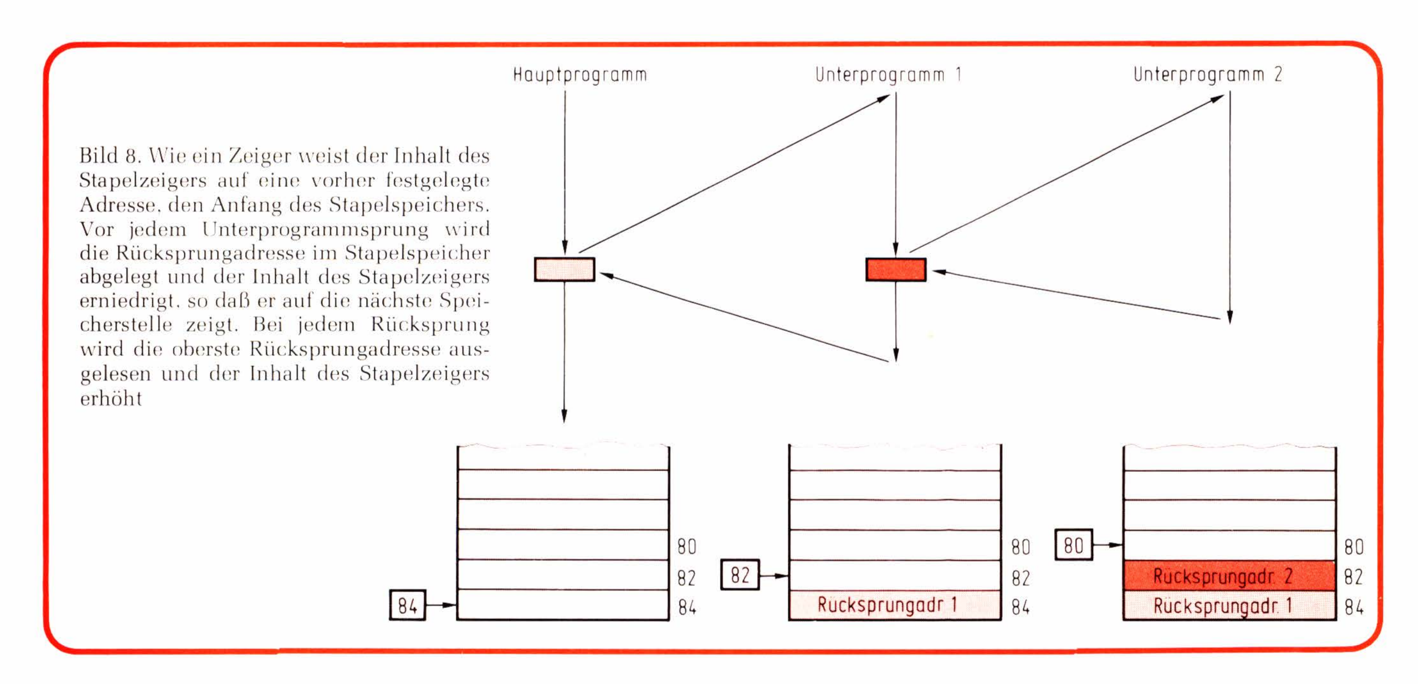


Bild 8. Wie ein Zeiger weist der Inhalt des Stapelzeigers auf eine vorher festgelegte Adresse, den Anfang des Stapelspeichers. Vor jedem Unterprogrammsprung wird die Rücksprungadresse im Stapelspeicher abgelegt und der Inhalt des Stapelzeigers erniedrigt, so daß er auf die nächste Speicherstelle zeigt. Bei jedem Rücksprung wird die oberste Rücksprungadresse ausgelesen und der Inhalt des Stapelzeigers erhöht

ration ebenfalls beteiligt werden. Wichtig sind diese Befehle vor allem für die Multiplikation und Division.

- *Arithmetische Befehle:* Vorhanden sind meist nur „Addition" und „Subtraktion". Zur Darstellung der negativen Zahlen verwendet man das sogenannte Zweierkomplement, bei dem man jedes Byte als ganze Zahl mit Vorzeichen deutet (höchste Stelle 1 = negativ). Die Dezimalzahl –1 ist dabei 1111 1111, die Dezimalzahl –2 ist 11111110 usw.

- *Logische Befehle:* Die wichtigsten sind UND- und ODER-Verknüpfungen. Eine UND-Verknüpfung wird beispielsweise nach folgendem Muster durchgeführt:

Operand A steht im Akkumulator	1101 00
Operand B steht im Speicher	0001 00
Ergebnis steht im Akkumulator	0001 00

d.h., die UND-Verknüpfung wird Bit für Bit durchgeführt. Auf diese Weise kann man z. B. beliebige Bits auf Null setzen, während man andere unverändert läßt (in unserem Beispiel bleibt das fünfte Bit des Akkumulators unverändert). Operand B wird oft auch als Maske bezeichnet.

- *Sprungbefehle:* Man unterscheidet zwischen unbedingten und bedingten Sprungbefehlen. Erstere laden in jedem Fall den Programmzähler mit der Adresse des Sprungziels, letztere tun das, wenn bestimmte Bits im Bedingungsregister gesetzt sind. Der Befehl „Springe zum Unterprogramm" veranlaßt zusätzlich die Abspeicherung der Rücksprungadresse im Stapelspeicher.

Darstellung von Befehlen

Das Programm steht als Folge von Befehlen im Speicher, und zwar in binärer Form. Da diese Darstellungsweise sehr unübersichtlich ist, faßt man jeweils vier Stellen einer Dualzahl zu einem Zeichen zusammen. Man benötigt dafür 16 verschiedene Zeichen (deshalb spricht man vom Sedezimalsystem – häufiger wird es allerdings als Hexadezimalsystem bezeichnet, was falsch ist). Die *Tabelle* zeigt, welche Zeichen den Dualzahlen 0000 bis 1111 zugeordnet sind.

Es muß aber noch einmal betont werden: Der Mikroprozessor kennt kein A, sondern nur die Bitkombination 1010, die andere Schreibweise ist nur eine Hilfe für den Programmierer.

Die meisten Mikroprozessoren kennen 1-Byte-, 2-Byte- und 3-Byte-Befehle. Da wir jedes Byte als zwei Sedezimalzahlen schreiben können, würde ein 2-Byte-Befehl etwa A6 0F lauten (binär: 1010 0110 0000 1111). Das erste Byte – der sogenannte Operationsteil – enthält immer die Information, was zu tun ist und aus wieviel Bytes der gesamte Befehl besteht. Das zweite und – falls vorhanden – das dritte Byte (Adreßteil) geben an, welche Operanden zu verwenden sind. In unserem Beispiel könnte A6 bedeuten: „Lade den Akkumulator mit dem Inhalt der Adresse, die im zweiten Byte des Befehls angegeben ist." Der Akkumulator würde also mit dem Inhalt der Adresse 0F geladen werden (die Adressen werden ebenfalls im Sedezimalsystem angegeben).

Dezimal	Dual	Sedezimal
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

Diese Befehle versteht der Mikroprozessor unmittelbar, wir sprechen deshalb von Maschinensprache.

Die sedezimale Schreibweise für die Befehle wird von den Mikroprozessor-Herstellern auf mitgelieferten Listen angegeben. Auf diesen Listen steht für jeden Befehl auch eine Abkürzung, die meist aus dem Englischen stammt und in etwa den Sinn des Befehls wiedergibt. Man nennt diese „Kürzel" auch mnemonischer Code, da sie leicht im Gedächtnis haften bleiben (Beispiel: LDA für *Load Accumulator*).

Wird ein Programm im mnemonischen Code erstellt, dann muß es auf jeden Fall in die Maschinensprache „übersetzt" werden. Man kann das von Hand mit der besagten Liste machen oder mit einem Computer. Der übersetzende Computer kann sogar der Mikroprozessor sein, für den das Programm geschrieben wurde. Das entspechende Übersetzungsprogramm nennt man Assembler. Unglücklicherweise bezeichnet man diese „Kürzelsprache" ebenfalls als Assembler (oder Assemblersprache).

Schlußbemerkung

Der Zweck dieses Artikels war nicht eine lückenlose Einführung in das Gebiet der Mikrocomputer zu geben. Es war vielmehr beabsichtigt, einen kleinen Einblick in deren Arbeitsweise zu vermitteln. Wer sich als Laie mit Computern beschäftigen will, sollte wissen, daß er Begriffe wie Stapelspeicher, Akkumulator, Bit und Byte gleich wieder vergessen kann – vorausgesetzt, er hat eine höhere Programmiersprache wie BASIC zur Verfügung. Will man sich in die Maschinensprache, z. B. mit Hilfe eines sogenannten Kits, einarbeiten, dann ist der beste Lehrmeister der Computer selbst. Also „ran an den Feind" – erst dann kommt auch gute Literatur zum Tragen. Das soll aber nicht heißen, daß man sich jetzt den nächstbesten Kit zulegen soll. Doch Informationsquellen gibt es genug – unter anderem in diesem Heft.

*) 1024 Byte = 1 KByte; das große K (Kilo) bedeutet in der Datentechnik eine Multiplikation um den Faktor 1024. Oft wird eine Speichergröße z. B. als 2 K angegeben. Unklar ist dabei zunächst, ob es sich um 2 KBit oder 2 KByte handelt. Klein k ist in diesem Zusammenhang falsch!



ommodore weist die Richtung:

Ver den Computer bisher aus Kostengründen scheute, ann ihn sich jetzt aus Kostengründen leisten!

hon eine knappe Übersicht der wichtigsten chnischen Daten macht deutlich: Commodore ET 2001 bietet ein Preis-Leistungsverhältnis, e man es noch vor kurzem für undenkbar hielt. rund hierfür ist die bahnbrechende Mikroozessor-Technik von Commodore und natürh die betont servicefreundlichen Steckmodule.

itgeliefert wird ein Bedienungshandbuch mit sführlicher Programmieranleitung. Eine Lernssette für die Computersprache Basic ist ebenls erhältlich.

rvice: Speziell für PET 2001 steht ein Händlerd Servicenetz bereit. Ein PET-Anwender-Club schon gegründet.

Leistungsdaten (Kurzfassung):
Arbeitsspeicher von 8 kByte bis zu 32 kByte zu erweitern. Insgesamt 312 Zeichen. 23 cm Bildschirm. Interface-Anschluß bis zu 15 Buskompatibler Geräte (IEC-Bus). Festwertspeicher mit 8 kByte Basic, 4 kByte Betriebssystem, 1 kByte Testroutine. Zubehör: 2. Kassette (Drucker 80 Zeichen breit, Floppy-Disk und Modem in Vorbereitung). 9 Standard- und 12 erweiterte Basic-Anweisungen. 12 math. Funktionen. 8 String (Zeichenketten)-Funktionen. 6 System-, 3 Formatierungs- und 4 Maschinensprache-Anweisungen. 3 logische Operatoren. 9 INPUT/OUTPUT-Anweisungen. Einfache-, Ganzzahl- und String-Variable u. v. m. Gewicht ca. 20 kg. 220 Volt. Maße: 42 x 47 x 36 cm.

Info- und Bestell-Coupon

- ○ Ich möchte zunächst ausführliches Informationsmaterial haben, sowie Bezugsquellennachweis
- ○ Ich bestelle 1 Stck. Commodore PET 2001 zum Preis von DM 2.900,– (inkl. MwSt.) – Verrechnungsscheck ist beigefügt –
- ○ Ich bin außerdem an der Mitgliedschaft im PET-Anwender-Club interessiert

(Bitte ankreuzen. Einsenden an Commodore)

Name: ____________________

Anschrift: ____________________

Unterschrift: (nur bei Bestellung) ____________________ h

commodore
Commodore GmbH · Frankfurter Str. 171-175
6078 Neu-Isenburg · Tel.: (06102) 8003 · Telex: 4185663 como d

Das Dualsystem

Uns allen geläufig ist das Dezimalsystem (Zehnersystem). Sehen wir uns mal die Zahl 3498 an. Wir können sie gedanklich von rechts nach links so zerlegen:

3	4	9	8	oder
			$8 \cdot 1$	8
		$9 \cdot 10$		$9 \cdot 10$
	$4 \cdot 100$			$4 \cdot 10 \cdot 10$
$3 \cdot 1000$				$3 \cdot 10 \cdot 10 \cdot 10$

d.h., jede weitere Stelle bedeutet eine Multiplikation mit 10. Man kann nun $10 \cdot 10 \cdot 10$ ebenso als 10^3 (sprich 10 hoch 3) $10 \cdot 10$ als 10^2 schreiben usw. – man nennt das Potenzschreibweise. Damit das ganze folgerichtig nach unten fortgesetzt werden kann, haben die Mathematiker festgelegt, daß $10^0 = 1$ gilt.
Und nicht nur das: Jede Zahl hoch Null ist Eins.

Also:
$3498 = 3 \cdot 10^3 + 4 \cdot 10^2 + 9 \cdot 10^1 + 8 \cdot 10^0$

Im Prinzip ist das alles im Zweiersystem genauso. Der Unterschied besteht darin, daß
1) statt 10 verschiedener Ziffern (0...9) nur zwei existieren, nämlich 0 und 1.
2) statt der Potenzen von 10 die Potenzen von 2 stehen.

Die Dualzahl 10101 hat also den dezimalen Wert 21:

1 0 1 0 1

$1 \cdot 2^0$	$1 \cdot 1$	1
$+ 0 \cdot 2^1$	$+ 0 \cdot 2$	$+ 0$
$+ 1 \cdot 2^2$	$+ 1 \cdot 2 \cdot 2$	$+ 4$
$+ 0 \cdot 2^3$	$+ 0 \cdot 2 \cdot 2 \cdot 2$	$+ 0$
$+ 1 \cdot 2^4$	$+ 1 \cdot 2 \cdot 2 \cdot 2 \cdot 2$	$+ 16$
		$= 21$

Die Umwandlung einer Dezimalzahl in eine Dualzahl kann man nach folgendem Schema vornehmen (es soll die Zahl 132 umgewandelt werden):

$132 : 2 = 66$ Rest 0
$66 : 2 = 33$ Rest 0
$33 : 2 = 16$ Rest 1
$16 : 2 = 8$ Rest 0
$8 : 2 = 4$ Rest 0
$4 : 2 = 2$ Rest 0
$2 : 2 = 1$ Rest 0
$1 : 2 = 0$ Rest 1

$= 1\,0\,0\,0\,0\,1\,0\,0$

Reinhard Gößler

Die Programmentwicklung für einen Mikrocomputer

Vom Hardware-Konzept bis zur PROM-Programmierung

Ein gut ausgebautes Mikrocomputer-Entwicklungssystem rangiert in der preislichen Größenordnung von 25 000...100 000 DM und liegt damit außerhalb aller Möglichkeiten eines Amateurs [1]. Aber man muß nicht unbedingt so hoch einsteigen, um einen eigenen Mikrocomputer zu entwerfen und zu programmieren. Wir wollen hier an einem konkreten Anwendungsbeispiel zeigen, wie eine solche Entwicklung auch mit sehr einfachen Mitteln zum Erfolg führen kann. Der Beitrag beschreibt die Vorgehensweise bei der Programmierung eines Mikrocomputers, der das Signal eines Zeitzeichensenders aufbereitet und die daraus gewonnene Information zur Anzeige bringt. Es wird im Detail gezeigt, welche einzelnen Schritte zur Realisierung dieser Aufgabe erforderlich sind. Prinzipiell unterscheidet sich diese Vorgehensweise in nichts von der professionellen Arbeit, nur wurde im vorliegenden Fall der Einsatz preiswerter Geräte in den Vordergrund gestellt.

1 Aufgabenstellung

Um die Aufgabenstellung für den Mikrocomputer definieren zu können, müssen die Randbedingungen bekannt sein. Dazu gehört beim hier betrachteten Beispiel das Prinzip der Informationsübertragung, um aus dem codierten Signal die Nutzinformation zurückzugewinnen.

1.1 Übertragungsprinzip

Die Ausstrahlung des amtlichen deutschen Zeitzeichens erfolgt über einen Langwellensender, dessen Steuerung die Physikalisch-Technische Bundesanstalt in Braunschweig übernimmt. Der Sender strahlt im Dauerstrich-Betrieb eine Trägerfrequenz aus, deren Amplitude im Sekundenrhythmus kurzzeitig abgesenkt wird. Dies ist die erste im Signal enthaltene Information, die den Sekundentakt überträgt. Zusätzlich dazu wird in codierter Form die komplette Zeitinformation ausgesendet, bestehend aus Angabe von Stunden und Minuten, Wochen- und Kalendertag, sowie Monat und Jahr. Diese Daten werden bitseriell im Verlauf einer Minute zusammen mit den Sekundenmarken übertragen. Um in diesem Telegramm die digitalen Zustände 0 und 1 unterscheiden zu können, macht man die Intervalle für die Trägerabsenkung unterschiedlich lang: Eine Absenkung von 100 ms Dauer entspricht dem digitalen 0-Zustand, und eine 200 ms dauernde Trägerabsenkung wird als logisch 1 interpretiert (*Bild 1*).

Der Wortbeginn fällt mit dem Minutensprung zusammen, also dem Sekundenwechsel von 59 auf 00. Um diesen Synchronisationszeitpunkt zu kennzeichnen, unterdrückt man den 59. Sekundenpuls. Das bedeutet, daß während jeder Minute die Trägerabsenkung nur während der Sekunden 0...58 vorgenommen wird. Das Ausbleiben eines Sekundenimpulses ist dann Kennzeichen dafür, daß zusammen mit dem nächsten Impuls die Übertragung eines neuen Zeittelegramms beginnt.

1.2 Informationscodierung

Zusammen mit den 59 Sekundenimpulsen können nach dem eben beschriebenen Schema 59 Bits zur Darstellung der Zeitinformation übertragen werden. Von diesen 59 möglichen werden nur 38 Bits ausgenutzt, und zwar diejenigen von den Sekunden-Nummern 21...58. Die Ziffern sind dabei im BCD-Code dargestellt, und auch der Wochentag erscheint als Zahl von 1...7; nach internationaler Übereinkunft entspricht die Eins dem Montag, fortlaufend bis zur Sieben, die für den Sonntag steht. Es ist übrigens müßig, den uralten Streit um diese Normung [2] und den „wahren" Anfang der Woche andauern zu lassen; denn erstens hat man sich glücklicherweise international auf die Lösung geeinigt, und zweitens haben Bibelexperten Zitate bereit, die unterschiedliche Tage als Wochenbeginn definieren können.

Die Zuordnung der Sekundenbits 21...58 zur Datums- und Zeitinformation zeigt *Tabelle 1*. Das Schema dieser Codierung ist gesetzlich verankert, so daß ein Aufbau zur Entschlüsselung und Anzeige dieser Information von bleibendem Wert ist. Die drei eingetragenen Prüfbits P1...P3 dienen zur Informationssicherung; sie ergänzen die jeweils vor ihnen liegenden Datenbits auf gerade Parität.

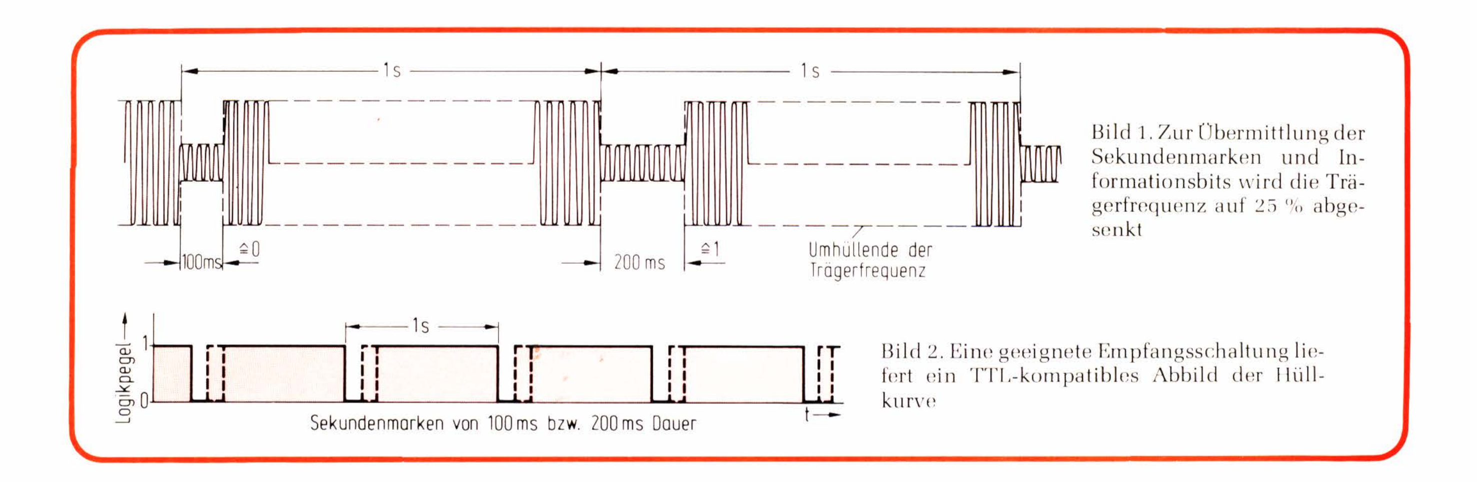


Bild 1. Zur Übermittlung der Sekundenmarken und Informationsbits wird die Trägerfrequenz auf 25 % abgesenkt

Bild 2. Eine geeignete Empfangsschaltung liefert ein TTL-kompatibles Abbild der Hüllkurve

1.3 Aufbereitung des Empfangssignals

Ehe der Mikrocomputer das analoge Empfangssignal verarbeiten kann, muß es verstärkt und seine Hüllkurve gleichgerichtet werden. Aufbau und Arbeitsweise einer geeigneten Empfängerschaltung sind nicht Gegenstand dieser Betrachtung. Wir wollen davon ausgehen, daß am Ausgang des Empfängers ein TTL-Signal bereitsteht, das die Hüllkurve des Eingangssignals abbildet (*Bild 2*). Bei den im Sekundenrhythmus auftretenden negativen Impulsen bleibt im Verlauf einer Minute ein Puls aus und signalisiert der Auswerteschaltung damit, daß die nächste Sekundenmarke der Beginn einer neuen Minute ist, wobei auch ein neues Zeittelegramm nach dem Schema der Tabelle 1 ausgesendet wird.

2 Konzeption des Mikrocomputers

Im Rahmen der Auswerteschaltung hat der Mikrocomputer die Aufgabe, das seriell ankommende Eingangssignal parallel auszuwerten und die entsprechende Information in einer Anzeige darzustellen. Es ist dies eine Problemstellung geringeren Umfangs, die sich ideal für den Einsatz eines Mikrocomputers eignet. Bereits an diesem einfachen Beispiel lassen sich komplexe Programmtechniken veranschaulichen, und die beim Mikrocomputer gegebene untrennbare Verschmelzung von Hard- und Software wird hierbei besonders deutlich.

2.1 Angepaßter Hardware-Aufbau

Oberstes Ziel des Hardware-Aufbaus war eine möglichst einfache und preiswerte Lösung. In entscheidendem Maße bestimmt diese Hardware-Organisation die spätere Programmerstellung. Prinzipiell ist jeder Mikrocomputer nach dem Schema Zentraleinheit, Speicher und Ein/Ausgabe-Leitungen aufgebaut (*Bild 3*). Die Zentraleinheit übernimmt die zentrale Ablaufsteuerung und Verwaltung sowie die Ausführung der arithmetischen und logischen Operationen. Im Speicher ist das Programm abgelegt (ROM-Speicher), das den Mikrocomputer bei schrittweiser Abarbeitung in die Lage versetzt, die gestellte Aufgabe auszuführen. Außerdem muß ein separater Arbeitsspeicher vorhanden sein (RAM-Speicher), um veränderliche Daten (z. B. die serielle Eingangsinformation oder Zwischenergebnisse) aufzunehmen bzw. bereitzustellen. Über die Ein/Ausgabe-Leitungen schließlich spielt sich der Informationsaustausch zwischen Computer und Umwelt ab. Auf das konkrete Anwendungsbeispiel bezogen heißt das zweierlei: Eingangsseitig muß ein Ein-bit-Kanal die vom Empfänger gelieferte Digitalinformation einlesen; und ausgangsseitig muß eine mehrstellige Anzeige geeignet angesteuert werden.

Augenscheinlich ist der Aufwand an der Ausgabeseite wesentlich höher als eingangsseitig. Um zu einer günstigen Hardware-Lösung zu kommen, muß deshalb besonderes Augenmerk auf den Ausgabekanal gelegt werden. Eine denkbare Lösung hierfür wäre der Einsatz eines programmierbaren E/A-Bausteins, der allerdings einige Nachteile mit sich bringt. Im allgemeinen bietet ein solcher Baustein weit mehr Leistungsfähigkeit als hier verlangt wird; resultierend daraus ist der Preis unverhältnismäßig hoch. Außerdem ist die Stromaufnahme derartiger Chips oft derart groß, daß die gesamte Leistungsbilanz verdorben wird.

2.2 Direkter Speicherzugriff

Eine sehr elegante Alternative zur programmgesteuerten Datenausgabe ist der Datenverkehr im direkten Speicherzugriff (engl. *direct memory access*, abgek. *DMA*). Dabei wird der Mikroprozessor zyklisch wiederkehrend kurz angehalten, und Daten- und

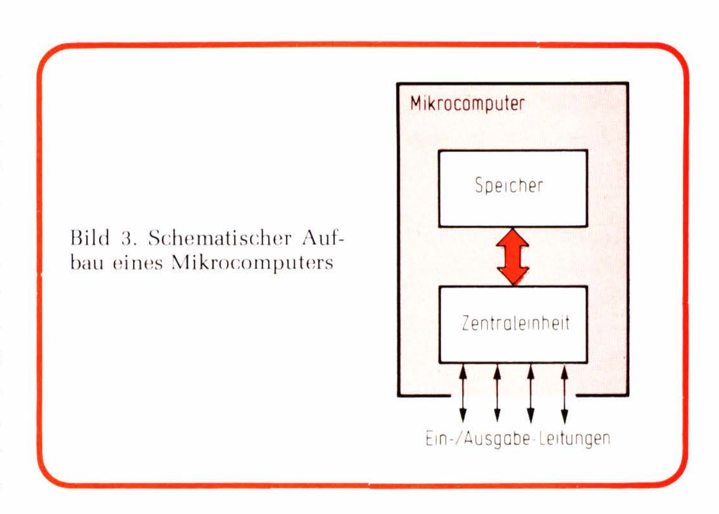


Bild 3. Schematischer Aufbau eines Mikrocomputers

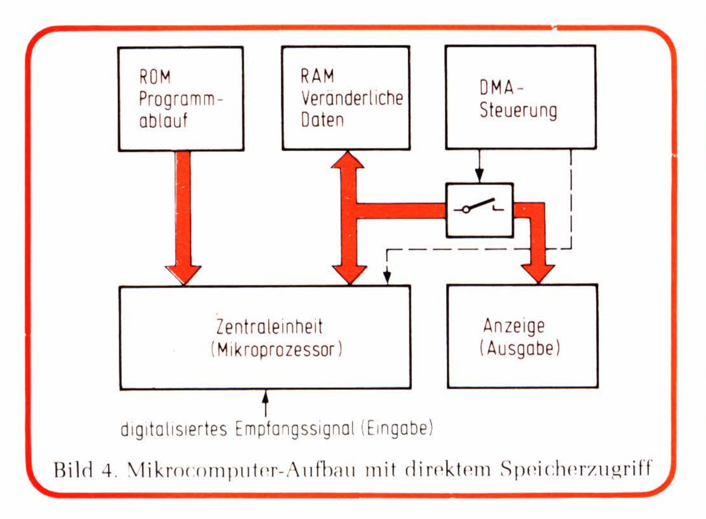


Bild 4. Mikrocomputer-Aufbau mit direktem Speicherzugriff

Adreßbus werden in den hochohmigen Zustand gebracht. Dann adressiert eine externe Stelle, die DMA-Steuerung, denjenigen Bereich des Arbeitsspeichers, aus dem die Information für die Anzeige ausgelesen werden soll. Nach dem Auslesen nimmt ein Auffangspeicher die Daten auf, und der Prozessor kann seine vorher unterbrochene Arbeit fortsetzen. Der ganze Vorgang spielt sich in der zeitlichen Größenordnung von wenigen Mikrosekunden ab und ist damit wesentlich schneller, als es eine programmgesteuerte Ausgabe sein könnte. Unter Ausnutzung des DMA-Verkehrs ergibt sich für den Mikrocomputer ein Aufbau, wie ihn *Bild 4* zeigt. Die DMA-Steuerung sorgt dafür, daß aus den obersten 16 Bytes des Arbeitsspeichers die Information ohne Einschaltung der Zentraleinheit ausgelesen und in der Anzeige dargestellt wird. Der Prozessor braucht folglich nur die entsprechenden Daten in diesen RAM-Bereich zu transportieren; ihr Auslesen erfolgt dann automatisch im DMA-Verkehr.

2.3 Mikroprozessor-Auswahl

Bei der Auswahl eines geeigneten Mikroprozessors sollten zwei allgemeine Gesichtspunkte im Vordergrund stehen: Erstens muß es nicht unbedingt ein 8080 sein, nur weil der vielleicht am populärsten ist. Und zweitens kann man einfachere Aufgaben nahezu mit jedem der gängigen Mikroprozessoren erledigen, und unter den eingangs erwähnten Voraussetzungen sollte der Preis hier die ausschlaggebende Rolle spielen. Wir haben uns deshalb für den Einsatz des Mikroprozessors SC/MP-II entschieden (*simple cost-effective microprocessor*), dessen Erstherstellер die amerikanische Firma National Semiconductor ist. Er wurde insbesondere für einfachere Anwendungen konzipiert und ist in mittleren Stückzahlen schon für weniger als 20.– DM erhältlich. In *Tabelle 2* sind einige charakteristische Merkmale dieses Prozessors zusammengestellt. Der Zusatz „Römisch-Zwei" in der Bezeichnung besagt, daß es sich gegenüber der ersten SC/MP-Ausführung um eine verbesserte Version handelt. Der Nachfolgetyp kommt beispielsweise mit einer einfachen 5-V-Versorgung aus, ist ansonsten aber Software-kompatibel mit der Erstversion.

Tabelle 1. Zuordnung der einzelnen Sekundenbits zur Datums- und Zeitinformation

Sek. Nr.	Wertigkeit	Information		Beispiel	
21	1			0	
22	2	Einer		1	2
23	4			0	
24	8		Minute	0	
25	10			1	
26	20	Zehner		0	5
27	40			1	
28	–	Prüfbit	P1	1	
29	1			1	
30	2	Einer		1	3
31	4			0	
32	8		Stunde	0	
33	10			0	2
34	20	Zehner		1	
35	–	Prüfbit	P2	1	
36	1			0	
37	2			0	0
38	4	Einer		0	
39	8		Kalender-	0	
40	10		tag	1	1
41	20	Zehner		0	
42	1		Wochen-	1	
43	2	Einer	tag	0	Montag
44	4			0	
45	1			1	
46	2	Einer		1	7
47	4			1	
48	8		Kalender-	0	0
49	10	Zehner	monat	0	
50	1			0	
51	2	Einer		0	8
52	4			0	
53	8		Kalender-	1	
54	10		jahr	1	
55	20			1	7
56	40	Zehner		1	
57	80			0	
58	–	Prüfbit	P3	1	

Beispiel: Montag, 10.07.78, 23.52 Uhr

Tabelle 2. Kurzbeschreibung des Mikroprozessors SC/MP II

8-bit-Datenbus
16 Adreßbits (64-KByte-Speicherumfang)
Integrierter Taktoszillator
Einzelne 5-V-Versorgung
Niedrige Stromaufnahme (max. 45 mA)
2 TTL-kompatible Eingänge
3 TTL-kompatible Ausgänge
Serieller Ein- und Ausgang
Möglichkeit des DMA-Verkehrs
Direkte Steuersignale für die Speicherverwaltung
Nur Ein- und Zwei-Byte-Befehle
Fünf Adressierungsarten

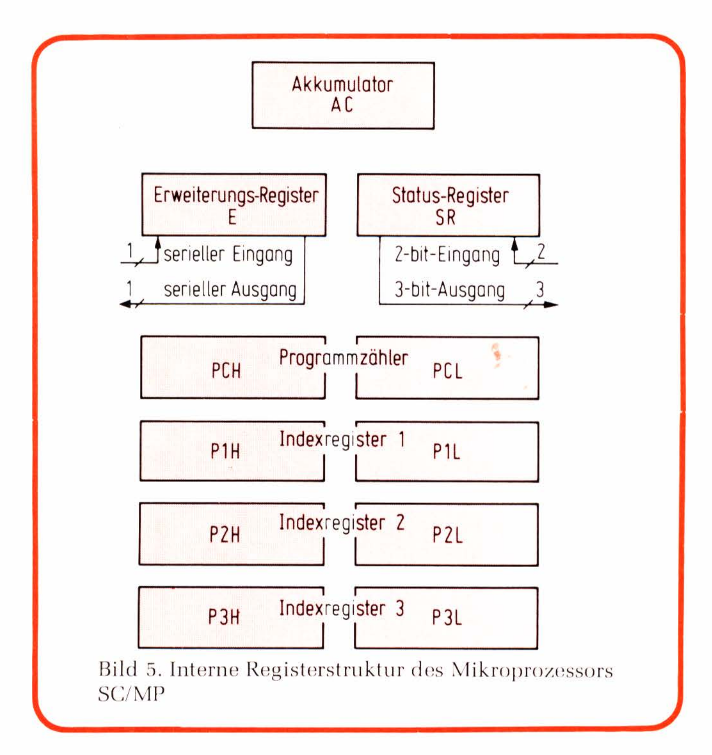


Bild 5. Interne Registerstruktur des Mikroprozessors SC/MP

2.4 Registerstruktur des SC/MP

Die interne Registerstruktur der Zentraleinheit ist in *Bild 5* dargestellt. Neben den drei 8-bit-Registern A, E und S sind vier 16-bit-Register vorhanden: Der Programmzähler PC und drei Indexregister P 1...P 3 (*pointer*). Die 16-bit-Register sind funktionell in ein unteres (*lower*) und ein oberes (*higher*) Byte aufgeteilt.

Der Akkumulator ist ein Universalregister, über das der Datenverkehr abläuft oder in dem einer von zwei Operanden gehalten wird. Hierhin gelangt auch stets das Ergebnis einer Operation.

Das Erweiterungsregister ist direkt mit dem Akkumulator verbunden. Hier kann bei Verknüpfung von Daten der zweite Operand stehen. Das oberste und unterste Bit dieses Registers sind als serieller Ein- bzw. Ausgang herausgeführt.

Den Aufbau des Statusregisters zeigt *Bild 6*. Die Bits 0...2 sind mit TTL-Pufferstufen herausgeführt, und in den Bits 4 und 5 erscheint diejenige Information, die an den zugehörigen Anschlußstiften anliegt. Diese fünf Bits sind also extern am Prozessor verfügbar, und man kann sie ohne zusätzlichen Aufwand als TTL-Ein- und Ausgabeleitungen einsetzen. Die Bits 6 und 7 werden entsprechend dem Ergebnis einer logischen oder arithmetischen Operation gesetzt, und das Bit 3 zeigt an, ob externe Programmunterbrechungen möglich sind oder ignoriert werden; dieses Bit wird softwaremäßig gesetzt bzw. gelöscht.

3 Die Software-Leistungsfähigkeit

Schon aus der Bezeichnung geht hervor, daß der SC/MP nur über eine begrenzte Software-Leistung verfügt. Darunter ist zu verstehen, daß der Befehlssatz nur einfachste Instruktionen enthält, die oft schon bei Operationen geringeren Umfangs ein mühsames Zusammensetzen erforderlich machen. Dennoch lassen sich mit entsprechendem Aufwand alle gewünschten Datenmanipulationen ausführen. Allerdings gehört der Aufruf von Unterprogrammen bei diesem Prozessor zum Umständlichsten, was man sich in der Mikrocomputertechnik vorstellen kann. Daß sich damit trotzdem komplexe Programme realisieren lassen, stellt einmal mehr die Vielseitigkeit von Mikroprozessoren unter Beweis.

3.1 Der SC/MP-Befehlssatz

Der vollständige Befehlssatz des SC/MP umfaßt die 46 Instruktionen, die in *Tabelle 3* zusammengestellt sind. Die erwähnte eingeschränkte Leistungsfähigkeit äußert sich beispielsweise darin, daß die Indexregister nur byteweise geladen werden können, was immerhin sechs Bytes im Programmspeicher belegt (vgl. Tabelle 6). Ein weiteres Beispiel hierfür ist das Fehlen eines Links-Schiebe-Befehls; wenn ein Register-Inhalt um ein Bit links verschoben werden soll, muß man das durch sieben Rechts-Schiebe-Befehle (RR) realisieren. Bei den Additionsbefehlen wird immer das CY/L-Bit mit einbezogen; in der Regel muß man also vor einer Addition dieses Bit löschen, um definierte Ausgangsbedingungen zu schaffen (vgl. Tabelle 10). Auf der anderen Seite gibt es aber im Befehlssatz auch Instruktionen, die in dieser Form kein anderer Mikroprozessor besitzt. Die Befehle der Gruppe 2 beispielsweise zählen den Inhalt einer Speicherstelle um Eins hoch (herunter) und laden den so entstandenen Wert in den Akkumulator; daß hierfür nur zwei Bytes im Programmspeicher erforderlich sind, ist recht günstig (die 8080-Familie kann das zwar auch mit nur zwei Byte erledigen, kostet dafür aber auch wesentlich mehr!). Besonders effektiv ist der Delay-Befehl aus der Gruppe 5; er ermöglicht auf einfache Weise eine programmierbare Zeitverzögerung, für die man im Befehl selbst und im Akkumulator die gewünschten Parameter vorgibt (vgl. Tabelle 7).

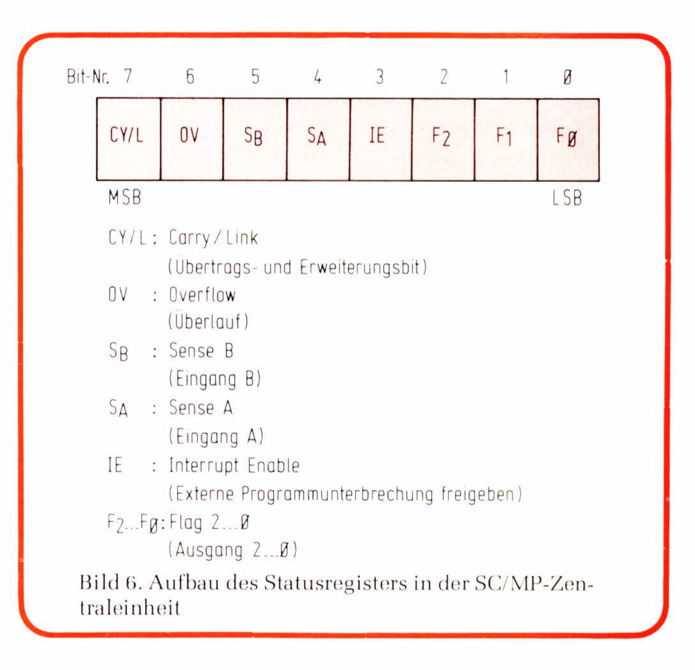


Bild 6. Aufbau des Statusregisters in der SC/MP-Zentraleinheit

Tabelle 3. Der vollständige SC/MP-Befehlssatz

Nr.	Befehls-gruppe	Mnemo-techn. Abk.	Beschreibung	Bedeutung	Bytes	Basis-Masch. Code	Befehlsaufbau erstes Byte 7|6|5|4|3	2	1|0	zweites Byte 7|6|5|4|3|2|1|0
1	Speicher-bezogene Befehle	LD	Load	Lade AC mit (EA)	2	C0				
		ST	Store	Lege (AC) in EA ab	2	C8				
		AND	AND	Logisch UND m. (AC) u. (EA)	2	D0				
		OR	OR	Logisch ODER m. (AC) u. (EA)	2	D8	Opcode	m	Ptr	Disp
		XOR	Exclusive-OR	Logisch EXOR m. (AC) u. (EA)	2	E0				
		DAD	Decimal Add	Dez. Add. (AC)+(EA)+(CY)	2	E8				
		ADD	ADD	Bin. Add. (AC)+(EA)+(CY)	2	F0				
		CAD	Complement and Add	(AC)+$\overline{\text{(EA)}}$+(CY)	2	F8				
2	Speicherstelle hochzählen/ erniedrigen	ILD	Increment and Load	AC u. (EA) m. (EA)+1 laden	2	A8	Opcode		Ptr	Disp
		DLD	Decrement and Load	AC u. (EA) m. (EA)-1 laden	2	B8				
3	Unmittelbare Befehle	LDI	Load Immediate	AC mit Byte 2 laden	2	C4				
		ANI	AND Immediate	Log. UND aus (AC) u. Byte	2		D4			
		ORI	OR Immediate	Log. ODER aus (AC) u. Byte 2	2	DC				
		XRI	Exclusive-OR Immediate	EXOR aus (AC) u. Byte 2	2	E4	Opcode			Daten
		DAI	Decimal Add Immediate	Dez. Add. (AC)+Byte 2+(CY)	2	EC				
		ADI	Add Immediate	Bin. Add. (AC)+Byte 2+(CY)	2	F4				
		CAI	Complement and Add Imm.	(AC)+$\overline{\text{Byte 2}}$ + (CY)	2	FC				
4	Sprungbefehle	JMP	Jump	Sprung zur EA	2	90				
		JP	Jump if Positive	Sprung zur EA, wenn (AC) 0	2	94				
		JZ	Jump if Zero	Sprung zur EA, wenn (AC) = 0	2	98	Opcode		Ptr	Disp
		JNZ	Jump if Not Zero	Sprung zur EA, wenn (AC) ≠ 0	2	9C				
5	Zeitverzöge-rungen	DLY	Delay	Zeitverzögerung von n Mikrozyklen $n = 13+2(AC)+2\,Disp+2^9\,Disp$	2	8F	Opcode			Disp
6	Register-E-Befehle	LDE	Load AC from Extension	AC mit (E) laden	1	40				
		XAE	Exchange AC and Extension	(AC) und (E) austauschen	1	01				
		ANE	AND Extension	Log. UND aus (AC) und (E)	1	50				
		ORE	OR Extension	Log. ODER aus (AC) und (E)	1	58	Opcode			
		XRE	EXOR Extension	EXOR aus (AC) und (E)	1	60				
		DAE	Decimal Add Extension	Dez. Add. (AC)+(E)+(CY)	1	68				
		ADE	Add Extension	Bin. Add. (AC)+(E)+(CY)	1	70				
		CAE	Complem. and Add Extens.	(AC)+$\overline{\text{(E)}}$+(CY)	1	78				
7	Modifikation der Index-register	XPAL	Exchange Pointer Low	(AC) und (PL) austauschen	1	30				
		XPAH	Exchange Pointer High	(AC) und (PH) austauschen	1	34	Opcode		Ptr	
		XPPC	Exchange Pointer with PC	(PC) und (Pointer) austauschen	1	3C				
8	Schiebebefehle	SIO	Serial Input/Output	Zykl. Verschieben von (E)	1	19				
		SR	Shift Right	Versch. v. (AC), 0 nachschieb.	1	1C				
		SRL	Shift Right with Link	Versch. v. (AC), (CY/L) nachsch.	1	1D	Opcode			
		RR	Rotate Right	Zykl. Verschieben von (AC)	1	1E				
		RRL	Rotate Right with Link	Zykl. Versch.v. (AC) und (CY/L)	1	1F				
9	Übrige Befehle	HALT	Halt	Halt-Flag aktivieren	1	00				
		CCL	Claer Carry/Link	CY/L-Bit löschen	1	02				
		SCL	Set Carry/Link	CY/L-Bit setzen	1	03				
		DINT	Disable Interrupt	Interrupt sperren (IE=0)	1	04	Opcode			
		IEN	Enable Interrupt	Interrupt freigeben (IE=1)	1	05				
		CSA	Copy Status to AC	AC mit (SR) laden	1	06				
		CAS	Copy AC to Status	SR mit (AC) laden	1	07				
		NOP	No Operation	PC erhöhen	1	08				

Bedeutung der Abkürzungen:

AC	Akkumulator
E	Register E
SR	Status-Register
EA	Effektive Adresse (siehe Adressierungsarten)
(AC)	Inhalt des Akkumulators
(EA)	Inhalt der durch die EA adressierte Speicherstelle
CY	Bit 7 im SR
IE	Bit 3 im SR
Ptr	Indexregister (Pointer)

3.2 *Adressierungsarten*

Neben der Aussage, *was* im einzelnen zu tun ist (Operation), muß ein Befehl immer angeben, *womit* etwas getan werden soll (Operand). Die Spezifikation des Befehls erfolgt immer im Feld „OpCode". Die übrigen Bits des Befehls geben eine Adresse an, unter der der betreffende Operand abgelegt ist. Diese Adresse wird beim SC/MP niemals als Nummer der betreffenden Speicherstelle eingesetzt. Eine solche Adressierungsart ist hier nicht vorgesehen, weil sie Drei-Wort-Befehle erfordern würde und der SC/MP-Befehlssatz nur Ein- und Zwei-Wort-Instruktionen vorsieht. Solange die Operanden-Adresse nicht unmittelbar aus dem Befehl hervorgeht (implizite oder unmittelbare Adressierung), errechnet sie sich aus verschiedenen Parametern, die im Befehl enthalten sind (Berechnung einer effektiven Adresse EA).

Die unterschiedlichen Adressierungsarten sehen nur auf den ersten Blick verwirrend aus. Deshalb sind im folgenden immer erläuternde Beispiele eingefügt, die den Sachverhalt klarstellen. In der Praxis lernt man diese Zusammenhänge völlig problemlos und sehr schnell kennen; man ist dann froh über die Leistungsfähigkeit, die sich aus der Vielfalt der Adressierungsmöglichkeiten ergibt. Zur Deutlichmachung der verschiedenen Adressierungsarten sollen Beispiele dienen, bei denen jeweils der Akkumulator auf unterschiedliche Weise mit Daten geladen wird.

3.2.1 Implizierte Adressierung *(implied)*

Die Adresse, unter der der Operand zu finden ist, braucht nirgendwo explizit angegeben zu werden, weil sie bereits aus dem Befehl selbst hervorgeht. Der Befehl CSA (Gruppe 9) lädt den Akkumulator mit dem Inhalt des Statusregisters SR. Ziel und Quelle des Datentransports sind implizit in der Wortstruktur dieses Ein-Byte-Befehls enthalten und werden bei der Befehlsdecodierung von der Zentraleinheit erkannt.

3.2.2 Unmittelbare Adressierung *(immediate)*

Der Operand ist als zweites Byte an den Befehl angehängt. Der Befehls-LDI (Gruppe 3) beispielsweise lädt diejenigen Daten in den Akkumulator, die im Byte nach dem Befehlscode „C4" stehen. Auch diese Form der Operanden-Spezifikation ist eine eigene Adressierungsart, obwohl man in der Literatur gelegentlich mißverständliche Angaben dazu findet. Nach der Decodierung des Befehlscodes C4 schaltet die Zentraleinheit nämlich die Adresse der nächsten Speicherstelle auf den Adreßbus, um das Datenbyte auszulesen; und dies ist selbstverständlich ein ganz normaler Adressierungsvorgang.

3.2.3 Relative Adressierung *(relative)*

Der Operand ist in einer Speicherstelle zu finden, deren Ort relativ zum augenblicklichen Programmzählerstand (PC) angegeben wird. Dieser „Adreßmode" gilt nur für Befehle der Gruppe 1, bei denen dann die Bits 0, 1 und 2 im Byte 1 auf 0 liegen müssen. Byte 2 des Befehls gibt als vorzeichenbehaftete Zahl (Zweierkomplement) an, um wieviel Speicherplätze verschoben, bezogen auf den aktuellen PC-Stand, der Operand zu finden ist. Diese Zahl wird als Versatz (*displacement*) interpretiert und kann die Werte −128...+127 annehmen. Wenn das Displacement-Feld auf −128 gesetzt wird, geht der Inhalt des Registers E als Versatz in die Adreßberechnung ein. Der Operand ist dann also so weit vom augenblicklichen PC-Stand verschoben, wie es die Zahl im Register E angibt. Wenn beispielsweise im Programmspeicher unter der Adresse 911 der Ladebefehl LD und in 912 der Wert 07 stehen, dann wird der Akkumulator mit demjenigen Datenbyte geladen, das +7 Speicherstellen entfernt steht.

3.2.4 Indizierte Adressierung *(indexed)*

Diese Adressierungsart benutzt zur Berechnung der effektiven Adresse immer eins der drei Indexregister (Pointer 1...3), zu dessen Inhalt der Versatz des Displacement-Feldes addiert (bzw. subtrahiert) wird. Die Angabe des Indexregisters erfolgt im Feld „Ptr" als Zwei-bit-Binärzahl; der Wert „00" im Ptr-Feld spezifiziert den Programmzähler, was im Abschnitt „Relative Adressierung" bereits beschrieben wurde. Wenn die indizierte Adressierung bei Befehlen der Gruppe 1 benutzt wird, unterscheidet das Bit 2 im Byte 1 (m-Feld) zwischen indizierter und selbst-indizierter Adressierung (Kennzeichnung, ob das Indexregister bei der Befehlsausführung modifiziert werden soll oder nicht). Im hier betrachteten Fall liegt dieses Bit auf Null.

Als Beispiel soll folgende Aufgabe ausgeführt werden: Lade den Akkumulator mit demjenigen Datenbyte, das um +7 Speicherstellen vom augenblicklichen Stand des Indexregisters 2 entfernt steht. Obwohl die Aussage in diesem Bandwurm-Satz klar definiert ist, soll sie noch einmal in ihren Bestandteilen betrachtet werden. Man tut gut daran, sich einen komplexen Sachverhalt auf diese Weise klarzumachen: Der Akkumulator soll mit einem Datenbyte geladen werden. Die Adresse für dieses Datenwort ist auf dem Umweg über das Indexregister 2 zu finden. Der Inhalt dieses Indexregisters zeigt auf eine Basis-Adresse (Pointer-Funktion). Das Displacement-Feld gibt an, wie weit das Datenbyte von der Basisadresse entfernt steht. Der Inhalt von Pointer 2 (Basisadresse) soll dabei unverändert bleiben (*Tabelle 4*).

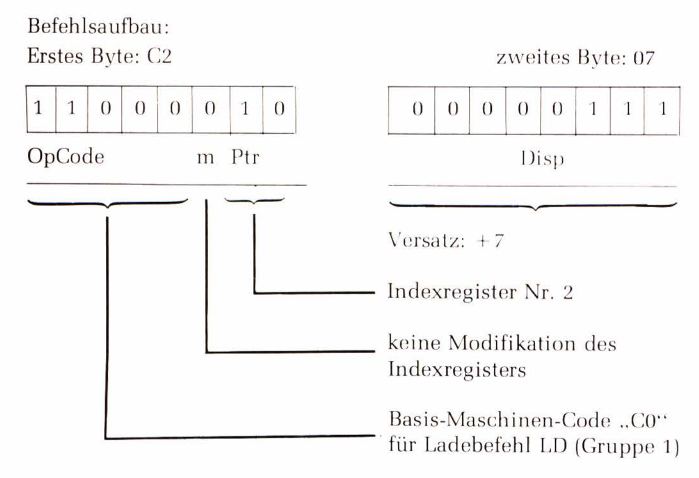
3.2.5 Selbst-indizierte Adressierung *(auto-indexed)*

Der Unterschied zur indizierten Adressierung besteht darin, daß das angesprochene Indexregister bei der Befehlsausführung modifiziert wird (m-Feld auf 1, nur bei Befehlen der Gruppe 1 möglich). Und zwar wird das Indexregister um den Wert des Displacement-Feldes erniedrigt, wenn dieser negativ ist; das geschieht *vor* dem Holen bzw. Ablegen des Datenbytes. Ist der Versatz positiv (oder Null), wird das Indexregister um diesen Wert erhöht, *nachdem* der Datentransport stattgefunden hat. Diese Organisation ist zur Verwaltung von Stapelspeichern (*stacks*) unentbehrlich.

Tabelle 4. Beispiel für die indizierte Adressierung

Aufgabe: Akkumulator AC mit Datenbyte laden, das unter der effektiven Adresse EA zu finden ist; die Basisadresse für EA ist im Indexregister 2 enthalten; das Datenbyte steht um +7 Speicherstellen davon entfernt.

Befehlsaufbau:
Erstes Byte: C2 zweites Byte: 07



4 Programmentwurf

In den Programmentwurf gehen verschiedene Randbedingungen mit ein, die entscheidend von der Hardware-Struktur bestimmt werden. Das Einlesen des digitalisierten Empfangssignals erfolgt beim vorliegenden Hardware-Aufbau über den Eingang B. Dieses Signal erscheint innerhalb der Zentraleinheit im Bit 5 (S_B) des Statusregisters und kann von dort aus weiterverarbeitet werden. Ausgangsseitig ist die Hardware so organisiert, daß die obersten 16 Speicherstellen des RAMs im direkten Speicherzugriff ausgelesen werden, um die Anzeige direkt anzusteuern. Zwischen diesem DMA-Puffer im RAM und den Siebensegment-Anzeigeeinheiten besteht hardwaremäßig der Zusammhang, den die *Bilder 7 und 8* zeigen. Der Prozessor muß also dafür sorgen, daß dieser RAM-Bereich mit denjenigen Daten geladen wird, die die empfangene Zeit- und Datumsinformation in der

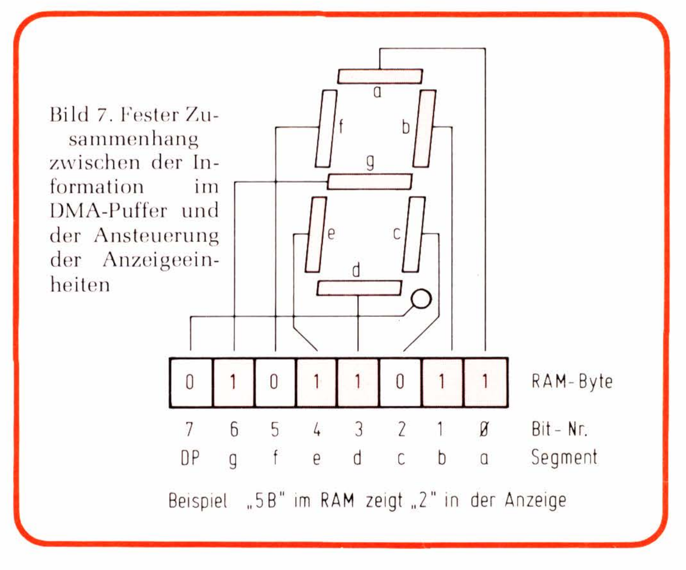


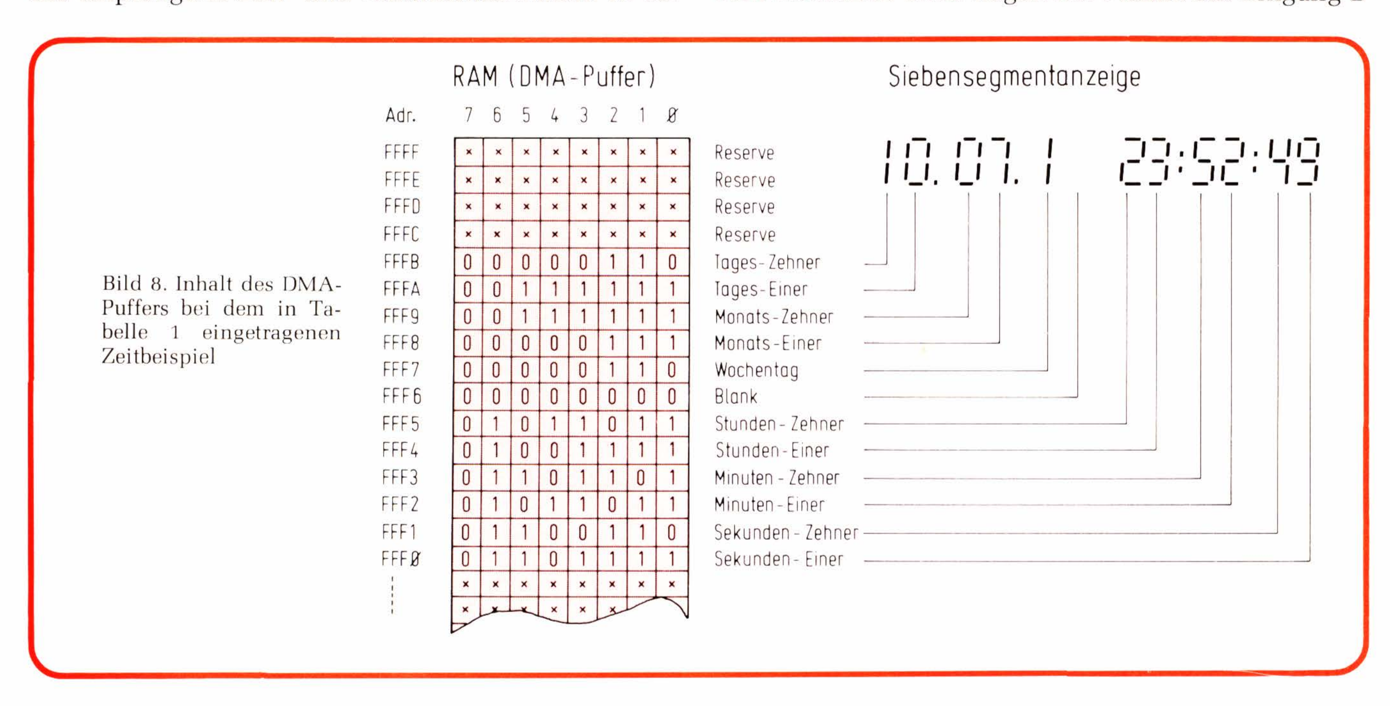
Bild 7. Fester Zusammenhang zwischen der Information im DMA-Puffer und der Ansteuerung der Anzeigeeinheiten

Anzeige erscheinen lassen. Auf die Hardware-Struktur, die das automatische Auslesen des DMA-Puffers mit anschließendem Multiplexen der Anzeige übernimmt, soll hier nicht näher eingegangen werden; Schaltungsdetails dazu sind in [3] zu finden.

4.1 Flußdiagramm

Entsprechend dem Blockschaltbild beim Hardware-Entwurf beginnt man die Programmerstellung mit dem Flußdiagramm, das einen generellen Ablaufplan beinhaltet. Um den Gesamtumfang in überschaubaren Grenzen zu halten, sollen weder das Flußdiagramm noch das Programm selbst bis ins Detail aufgelöst werden. Vielmehr sollen einzelne Beispiele die prinzipielle Vorgehensweise bei der Umsetzung einer gestellten Aufgabe in die Software zeigen.

Das Programm ist eine Endlosschleife, die nach dem Herstellen bestimmter Anfangsbedingungen (Initialisieren) fortwährend durchlaufen wird (*Bild 9*). Nach dem Eintreffen einer negativen Flanke am Eingang B



Adr.	7	6	5	4	3	2	1	Ø	
FFFF	×	×	×	×	×	×	×	×	Reserve
FFFE	×	×	×	×	×	×	×	×	Reserve
FFFD	×	×	×	×	×	×	×	×	Reserve
FFFC	×	×	×	×	×	×	×	×	Reserve
FFFB	0	0	0	0	0	1	1	0	Tages-Zehner
FFFA	0	0	1	1	1	1	1	1	Tages-Einer
FFF9	0	0	1	1	1	1	1	1	Monats-Zehner
FFF8	0	0	0	0	0	1	1	1	Monats-Einer
FFF7	0	0	0	0	0	1	1	0	Wochentag
FFF6	0	0	0	0	0	0	0	0	Blank
FFF5	0	1	0	1	1	0	1	1	Stunden-Zehner
FFF4	0	1	0	0	1	1	1	1	Stunden-Einer
FFF3	0	1	1	0	1	1	0	1	Minuten-Zehner
FFF2	0	1	0	1	1	0	1	1	Minuten-Einer
FFF1	0	1	1	0	0	1	1	0	Sekunden-Zehner
FFFØ	0	1	1	0	1	1	1	1	Sekunden-Einer
⋮	×	×	×	×	×	×	×	×	
	×	×	×	×	×	×			

Bild 8. Inhalt des DMA-Puffers bei dem in Tabelle 1 eingetragenen Zeitbeispiel

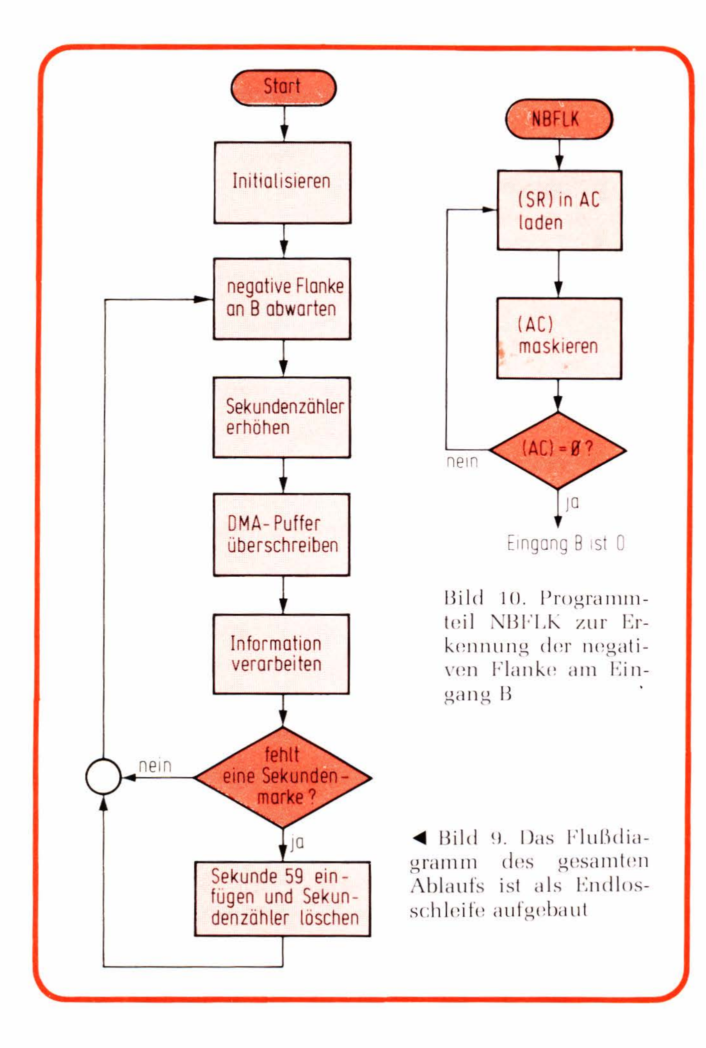


Bild 10. Programmteil NBFLK zur Erkennung der negativen Flanke am Eingang B

◀ Bild 9. Das Flußdiagramm des gesamten Ablaufs ist als Endlosschleife aufgebaut

wird ein Software-Zähler aktiviert, in dem die Sekunden-Nummer steht. Dieser Zählerstand wird erstens dazu benutzt, den Sekunden-Stand anzuzeigen; und zweitens geht daraus hervor, wie das gerade empfangene Bit weiterzuverarbeiten ist. Wie aus Tabelle 1 zu erkennen ist, gehört beispielsweise das bei Sekunde 31 eintreffende Bit zur Einerstelle der Stunde; innerhalb dieser BCD-Stelle besitzt es die Wertigkeit 4 und muß dementsprechend eingeordnet werden (Informationsverarbeitung). Da die negative Flanke an B den Sekundenwechsel markiert, soll noch vor der weiteren Verarbeitung der DMA-Puffer aktualisiert werden, um die neue Sekunde anzuzeigen.

Das Fehlen einer Sekundenmarke ist das Kennzeichen dafür, daß beim nächsten Eingangsimpuls eine neue Minute beginnt; gleichzeitig wird der Sekundenzähler gelöscht und läuft von da an wieder synchron mit. Der fehlende 59. Sekundenimpuls muß vom Computer eingefügt werden, damit das kontinuierliche Weiterzählen gewährleistet ist.

4.2 *Software-Elemente*

Das grobe Ablaufschema des Flußdiagramms muß nach und nach aufgeschlüsselt werden, um ein geeignetes Programm zu erstellen, das die betreffende Aufgabe ausführen kann. Diese Vorgehensweise soll an einigen typischen Beispielen erläutert werden.

4.2.1 Warteschleife

Den Beginn einer neuen Sekunde kennzeichnet die negative Flanke am Eingang B. Um diese zu erkennen, muß das Programm in einer Warteschleife verharren, bis ein 1/0-Übergang auftritt. Zur Abfrage der Eingangsleitung muß die Information aus dem Statusregister, wo sie von außen eintrifft, in den Akkumulator geladen werden (*Bild 10*). Die Abfrage, ob ein einzelnes Bit gesetzt ist, nimmt man über eine sogenannte Maskierung vor: Der Akkumulator-Inhalt wird mit einem solchen Datenbyte (Maske) UND-verknüpft, bei dem nur das abzufragende Bit auf 1 liegt. Wenn das Ergebnis Null ist, war das fragliche Bit nicht gesetzt, und im anderen Fall ist das Ergebnis der UND-Verknüpfung ungleich Null. Softwaremäßig erfolgt eine solche Abfrage mittels bedingtem Sprungbefehl; im hier betrachteten Fall lautet die Abfragebedingung auf den Nullzustand des Akkumulators.

Die Umsetzung dieser Aktivität beansprucht fünf Byte im Programmspeicher (*Tabelle 5*). Die für die UND-Verknüpfung benutzte Maske „20" hat im Bit 5 eine 1 und maskiert damit genau das Bit, in dem die Information vom Eingang B steht. Der bedingte Sprung wird nur dann ausgeführt, wenn der Akkumulator-Inhalt ungleich Null ist. In diesem Fall lag das maskierte Bit auf 1, und im Akkumulator steht nach Ausführung des ANI-Befehls das Ergebnis 20. Die Angabe des Sprungziels in diesem Befehl ist ein erstes konkretes Beispiel für eine Adreßberechnung. Am einfachsten läßt sich in diesem Fall die Sprungweite relativ zum PC-Stand angeben (Bits 0 und 1 im Sprungbefehl auf 0): Der Programmzähler muß um −5 zurückgestellt werden, um an die Programmstelle NBFLK zu gelangen; das Auszählen der Sprungweite geschieht vom nächstfolgenden Befehl aus, weil der Programmzähler bereits vor der Ausführung eines Befehls erhöht wird. Als Versatz muß ins Displacement-Feld also die negative Fünf als Zweierkomplement eingesetzt werden. Das Zweierkomplement einer Digitalzahl erhält man durch Invertieren und anschließendes Hochzählen um Eins: Das Invertieren der 8-bit-Zahl 05 ergibt FA (0000 0101 → 1111 1010), und das Erhöhen führt zu FB, was als −5 interpretiert wird.

4.2.2 Serien/Parallel-Umsetzung

Die seriell eintreffenden Bits müssen erst decodiert und dann parallel zusammengefaßt werden. Die Erkennung, ob das mit einer Sekundenmarke zusammen übertragene Bit eine Null oder Eins ist, geht sehr ein-

Tabelle 5. Programmteil NBFLK

Masch.-Code	Marke	Assembler-Code	Kommentar
06	NBFLK	CSA	(SR) in den Akku laden
D4		ANI 20	Bit 5 maskieren
20			
98		JNZ NBFLK	Sprung zum Anfang,
FB			wenn Ergebnis ungleich Null

fach vor sich (*Bild 11*): Wenn man ungefähr 150 ms nach der negativen Flanke an B den Zustand des B-Eingangs erneut abfragt, erhält man bei einer 100-ms-Marke eine 1 und bei einer 200-ms-Marke eine 0; das ist genau das Inverse der eigentlichen Bedeutung, so daß man den nach 150 ms eingelesenen Wert nur zu invertieren braucht.

Das Sammeln der einzelnen Bits und das Zusammenfassen zu einer parallelen Information geschieht nach dem in *Bild 12* angegebenen Schema. Eine Speicherstelle im RAM fungiert dabei als Hilfsregister HR, in dem die seriell eintreffende Information zu einer BCD-Zahl zusammengefaßt wird. Auf die Adresse dieser RAM-Zelle ist vorher ein Pointer eingestellt worden, um anschließend mittels indizierter Adressierung die Inhalte von Akkumulator und Hilfsregister miteinander ODER-zuverknüpfen. Dabei wird zum alten Inhalt von HR ein Bit hinzugefügt, und die dadurch entstandene Information wird um ein Bit nach rechts verschoben, ehe sie als neuer Wert im HR abgelegt wird. Das entsprechende Programm (*Tabelle 6*) benutzt das Indexregister 2 zur Adressierung, und das Hilfsregister wird in Speicherstelle 2606 eingerichtet.

4.2.3 Programmierte Zeitverzögerungen

Für das Einlesen der aktuellen Information vom Eingang B war die Forderung aufgetreten, dies 150 ms nach der negativen Signalflanke zu veranlassen. Eine solche Zeitverzögerung kann beim SC/MP problemlos mit dem Delay-Befehl (Gruppe 5) realisiert werden. Allerdings ist der Prozessor während der gesamten Wartezeit beschäftigt und kann dabei keine anderen Aufgaben übernehmen.

In die Angabe für die Zeitverzögerung gehen folgende zwei Parameter mit ein (vgl. Tabelle 3): der Akkumulator-Inhalt (AC) und der Wert im Displacement-Feld (DISP) des DLY-Befehls. Nach der Formel

$$n = 13 + 2 \cdot |257 \cdot (DISP) + (AC)|$$

verbringt der Prozessor dann n Mikrozyklen mit der Ausführung dieses Befehls, wobei ein Mikrozyklus

Tabelle 6. Programmteil SRPAR mit vorhergehendem Setzen des Pointers 2 (SETP 2)

Masch.-Code	Marke	Assembler-Code	Kommentar
C4	SETP2	LDI 06	unteres Adreßbyte für
06			Pointer 2 laden
32		XPAL 2	(AC) nach P2L laden
C4		LDI 26	oberes Adreßbyte für
26			Pointer 2 laden
36		XPAH	(AC) nach P2H laden
nach dem Einlesen des Eingangs B:			
DA	SRPAR	OR	ODER-Verknüpfung
00			von (AC) und (2606);
			(über P 2 adressiert)
1C		SR	(AC) rechts verschieben
CA		ST	(AC) nach 2606 laden
00			(über P 2 adressiert)

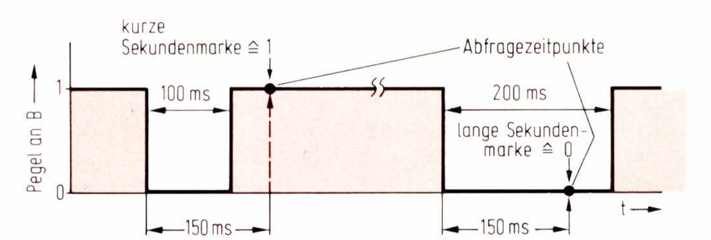


Bild 11. Der Abfragezeitpunkt liegt 150 ms hinter der negativen Flanke an B

aus vier Taktperioden besteht. Nimmt man einmal eine Taktfrequenz von 1 MHz an, wäre ein Mikrozyklus 4 µs lang, und mit $(AC) = 240_{10}$ und $(DISP) = 72_{10}$ ergäbe sich (ungefähr) die geforderte Wartezeit von 150 ms. *Tabelle 7* zeigt den entsprechenden Programmteil, der diese Zeitverzögerung erzeugt. Es werden dafür lediglich vier Byte im Programmspeicher belegt, und das bedeutet eine wirklich optimale Lösung.

4.2.4 Programmverzweigungen nach Tabelle

Bei der Weiterverarbeitung der seriell eingelesenen Daten besteht die Aufgabe, diese nach dem Schema der Tabelle 1 zu sortieren und geordnet abzulegen. Während bei den Sekundenpulsen 21, 22 und 23 beispielsweise nur jeweils das eingelesene Bit ins Hilfsregister nachgeschoben zu werden braucht, muß beim Impuls Nr. 24 außerdem die dann komplettierte BCD-Zahl noch abgespeichert werden. Die Reaktion des Programms ist also je nach Sekunden-Nummer unterschiedlich, zumal manche Digits auch nur aus 2 oder 3 Bits bestehen (Stunden-Zehner, Wochentag). Um an die entsprechenden Stellen im Programm zu verzweigen, könnte man nun endlos viele Abfragen hintereinander anordnen, um den Stand des Sekundenzählers zu ermitteln. Das wäre allerdings nicht nur unelegant, sondern es würde auch noch einen riesigen Programmaufwand erfordern.

Wesentlich einfacher und eleganter ist es, wenn man den Computer in einer Tabelle „nachsehen" läßt, was er beim jeweiligen Stand des Sekundenzählers im einzelnen tun soll. Wie das konkret abläuft, ist in den *Tabellen 8 und 9* an einem praktischen Beispiel zusammengestellt. Ausgangspunkt dabei ist eine Speicherorganisation, wie sie auszugsweise in Tabelle 8 wiedergegeben ist. Jeden der drei dort skizzierten Speicherbereiche spricht ein eigenes Indexregister an, das dazu entsprechend geladen werden muß (vgl. Tabelle 9, Vorbereitungen).

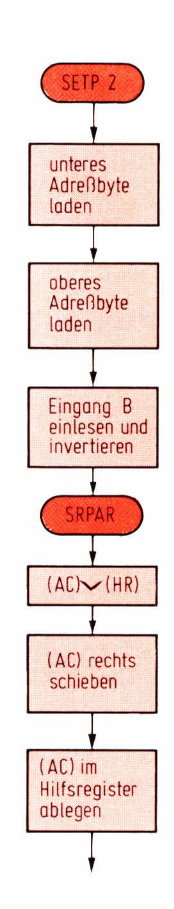


Bild 12. Bevor die Serien/Parallel-Umsetzung SRPAR aufgerufen werden kann, muß P2 geladen und Eingang B abgefragt werden

Der Sekundenzähler soll in diesem Beispiel gerade den dezimalen Stand 32 (sedezimal 20) erreicht haben (vgl. Tabelle 9, Ausgangspunkt); das ist der Zeitpunkt, zu dem das letzte Bit des Stunden-Einers eingelesen ist (vgl. Tabelle 1). Danach soll das Programm an diejenige Speicherstelle springen, an der dieses Stunden-Einer-Digit weiterverarbeitet wird (im Beispiel ist das die Adresse 3087). Wie diese Programmverzweigung auf dem Umweg über die Sprungtabelle abläuft, zeigt in den einzelnen Phasen die Tabelle 9. Das zugehörige Programm ist in *Tabelle 10* aufgelistet; es umfaßt nur wenige Bytes, zu denen allerdings noch die Tabelle mit den Sprungadressen hinzukommt.

Auf diese Weise lassen sich auch alle anderen Umsetzungen durchführen, was z. B. bei der Code-Umwandlung (Binär- in Siebensegmentcode) zur Ansteuerung der Anzeige erforderlich ist.

5 Programmtest

Nach dem Entwurf eines Programmteils muß man auf irgendeine Weise verifizieren, ob die beabsichtigten Eigenschaften vom Programm auch ausgeführt werden. Man kann davon ausgehen, daß der erste Entwurf eines Programms niemals die endgültige Version darstellt. In aller Regel sind zunächst noch Fehler enthalten, die die verschiedensten Ursachen haben. Im Aufspüren und Erkennen dieser Fehler liegt die eigentliche Schwierigkeit, die mit der Mikrocomputertechnik verbunden ist. Verglichen damit sind der Hardware-Entwurf und die Programmierung fast unproblematisch.

Tabelle 7. Programmteil DLY15 zur Erzeugung einer 150-ms-Zeitverzögerung

Masch.-Code	Marke	Assembler-Code	Kommentar
C4	DLY15	LDI F0	AC mit 240_{10} laden
F0			
8F		DLY	DISP = 72_{10}
48			

Tabelle 8. Speicherorganisation

Adresse	Inhalt
2600 ff	Hilfsregister und Zähler im RAM
2606	Hilfsregister HR zur Serien/Parallel-Umsetzung
2607	Sekundenzähler
3000 ff	Programmteile zur Informationsverarbeitung
3087 ff	Programmteil zur Verarbeitung des Stunden-Einers
5249 ff	Tabelle mit Sprungadressen zu den Programmteilen
5269	unteres Adreßbyte des Programmteils, der den Stunden-Einer verarbeitet (87)

Der Grund für diese Schwierigkeiten ist in der Unzulänglichkeit der vorhandenen Meßmittel zu suchen. Bezogen auf die Hardware bedeutet das, daß herkömmliche Meßgeräte wie Oszillograf oder Logikprüfstift allein nicht ausreichen, um ein Mikrocomputer-System meßtechnisch zu erfassen. Durch die ineinander verschachtelten Zeitabläufe und den bidirektionalen Informationsfluß auf vielen Leitungen erfordert die Signalverfolgung Spezialgeräte, wie z. B. einen Logikanalysator. Softwaremäßig liegt die Problematik bei Unterstützungsprogrammen (Monitor-Programmen), mit deren Hilfe man Fehler in eigenen Programmen einkreisen kann. Auf beiden Gebieten wird wohlgemerkt jede nur denkbare Unterstützung angeboten, die allerdings wegen der damit verbundenen Kosten außerhalb des Hobby-Budgets liegt.

Dennoch hat auch der Amateur die Möglichkeit, einen eigenen Mikrocomputer zu entwickeln und diesen zu programmieren. Fehlende Spezialgeräte muß man dann durch erhöhten Arbeitsaufwand und durch kombinierendes Nachdenken ersetzen. Eins aber steht unumstößlich fest: Mit dem entsprechenden Einsatz und der nötigen Zielstrebigkeit kommt man auch auf diesem komplexen Gebiet zum Ziel.

Der effektivste Weg für das Austesten von Programmen im Prototypen-Aufbau besteht in der ROM-Simulation. Dabei ist der arbeitsfähige Mikrocomputer-Aufbau bereits vorhanden, und an die Stelle des späteren Programmspeichers wird als ROM-Ersatz ein RAM eingesetzt, das von außen her geladen werden kann. Wenn sich in diesem Simulationsspeicher die zu testenden Programmteile befinden, übergibt man die Speicherverwaltung an den Prozessor im Prototyp. Dieser arbeitet dann mit dem Simulationsspeicher genauso wie mit dem später dafür eingesetzten ROM-Programmspeicher. Nur läßt sich das zur Simulation verwendete RAM bei Programmfehlern direkt und ohne komplizierte Löschvorgänge ändern. Auf diese Weise lassen sich selbst längere Programme Schritt für Schritt in Betrieb nehmen.

Der Simulationsspeicher läßt sich am einfachsten dadurch realisieren, daß man auf einen fertigen „Spielcomputer" zurückgreift, der heute bereits sehr

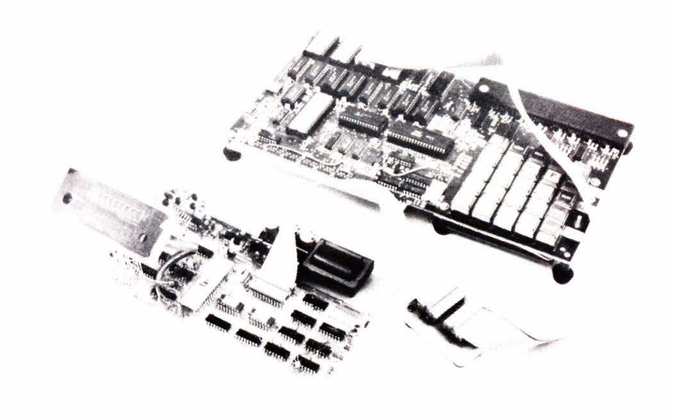


Bild 13. Der RAM-Bereich eines Einfach-Computers dient zur ROM-Simulation im Prototyp

Tabelle 9. Schema einer Programmverzweigung nach Tabelle

		(AC)	(P1)	(P2)	(P3)
Vorbereitungen:	P 3 halt eine Bezugsadresse für das RAM				26 00
	P 2H hält das obere Byte des Tabellenanfangs			52 XX	
	P 1H hält das obere Byte des Programmanfangs		30 XX		
Ausgangspunkt:	Sekundenzähler SZ hat den Stand $32_{10} = 20_{16}$ erreicht (letztes Bit des Stunden-Einers ist eingelesen)				
Ablauf:	(SZ) nach AC laden (über P 3)	20			
	Tabellenanfang (LOW) addieren: $+49_{16}$	69			
	(AC) nach P2L laden			52 69	
	Sprungadresse (LOW) aus der Tabelle holen (über P 2)	87			
	(AC) nach P1L laden		30 87		

preiswert erhältlich ist. Über die Tastatur lädt man die Befehle im Maschinencode in den Arbeitsspeicher und inaktiviert danach den Mikroprozessor des Simulations-Computers, dessen Daten- und Adreßbus mit dem Prototypenaufbau verbunden sind (*Bild 13*). Von diesem Augenblick an benutzt der selbst entwickelte Mikrocomputer das fremde RAM quasi als „Gasthörer" und behandelt es wie seinen eigenen Programmspeicher.

Der „Spielcomputer" hat wohlgemerkt nur die Aufgabe, das zur Simulation dienende RAM zu laden. Er muß dazu natürlich nicht mit demselben Mikroprozessor ausgestattet sein, den man beim Prototyp einsetzt.

6 PROM-Programmierung

Wenn diejenige Programmversion gefunden ist, die die gestellte Aufgabe befriedigend löst, wird man damit ein PROM laden wollen, um die Information permanent verfügbar zu haben. Sieht man einmal von der Möglichkeit eines maskenprogrammierten ROMs ab (die Maskenkosten liegen in der Größenordnung von $ 1000, aber noch zum alten Kurs!), bleiben drei Wege der PROM-Programmierung übrig: Verwendung bipolarer PROMs, UV-löschbarer oder elektrisch lösch- und wiederprogrammierbarer Typen.

Jeder PROM-Typ erfordert ein passendes Programmiergerät zum dauerhaften Einschreiben der Information. Derartige Geräte sind allerdings ziemlich teuer, aber man kann sich mit relativ geringem Aufwand eine passende Schaltung auch selbst aufbauen (*Bild 14*). In der Praxis ist das tatsächlich so einfach, wie es sich anhört: Um ein PROM zu programmieren, muß man nur im richtigen Augenblick am richtigen Anschlußstift die richtige Spannung anlegen. Die dazu notwendigen Informationen gehen detailliert aus den betreffenden Datenblättern hervor, und auch hierbei leistet ein Kleinst-Computer gute Dienste, um die Handhabung der Daten beim Programmieren zu übernehmen.

Für welchen ROM-Typ man sich im einzelnen entscheidet, hängt von verschiedenen Überlegungen ab; alle haben gewisse Vor- und Nachteile. Die bipolaren Typen lassen sich nur einmal programmieren. Spätere Änderungen sind nur insoweit möglich, als man die noch nicht programmierten Bits nachträglich „einbrennt". Dieser Vorgang ist nicht umkehrbar. Der Vorteil liegt in der Angebotsvielfalt und im günstigen Preis.

Die beiden anderen ROM-Kategorien haben den Vorteil, daß man ihren (permanenten) Inhalt löschen und neu programmieren kann. Es gibt die Möglichkeit, das Löschen mit kurzwelligem UV-Licht durch-

Tabelle 10. Programmteil SRPG zur Programmverzweigung nach einer Sprungtabelle

Masch.-Code	Marke	Assembler-Code	Kommentar
C3	SPRG	LD	Speicherstelle (P3+7) laden (Sekundenzähler in 2607)
07			
02		CCL	CY/L löschen (für die nachfolgende Addition)
F4		ADI 49	zu (SZ) Tabellenanfang addieren
49			
32		XPAL 2	(AC) nach P2L laden
C2		LD	Speicherstelle (P2+0) laden (Sprungadresse LOW)
00			
31		XPAL 1	(AC) nach P1L laden
3D		XPPC 1	(P1) und (PC) austauschen: Programmsprung zu der in (P1) enthaltenen Adresse

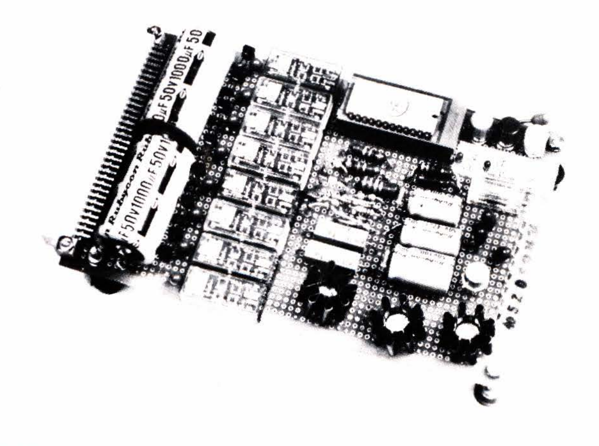


Bild 14. Laboraufbau eines Programmiergerätes für elektrisch löschbare PROMs

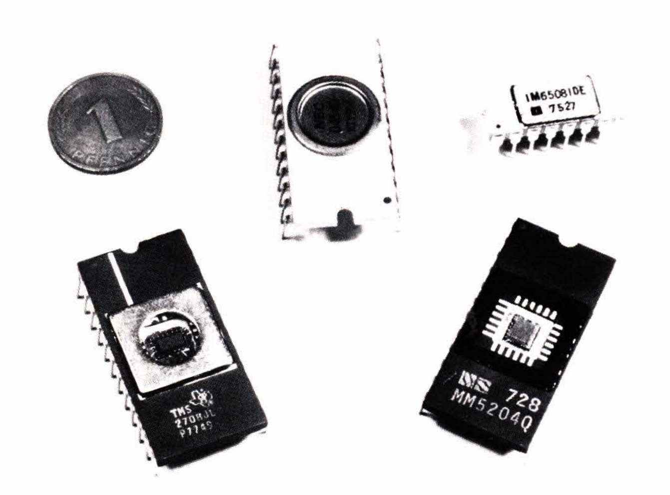


Bild 15. UV-löschbare PROMs sind an dem charakteristischen Fenster erkennbar

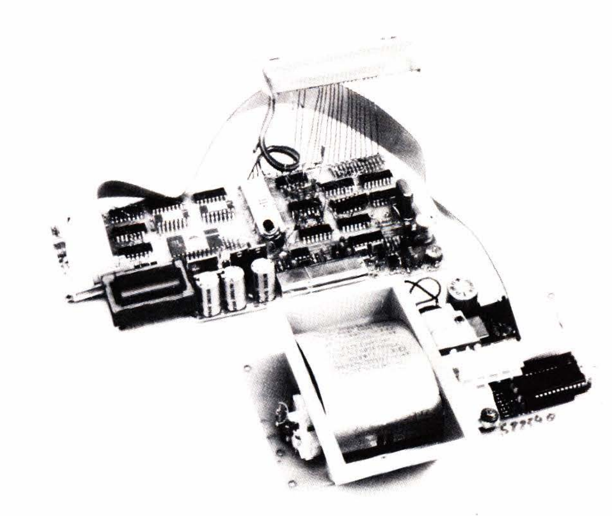


Bild 16. Einfache Drucker lassen sich ohne Anpassungsprobleme direkt vom Mikrocomputer ansteuern

zuführen; die dafür vorbereiteten PROMs haben dazu ein Sichtfenster, durch das das Licht einfallen kann (*Bild 15*). Obwohl der Löschvorgang technisch vollkommen unkritisch ist, sollte man gerade beim Umgang mit UV-Lampen äußerst vorsichtig sein (Verbrennungsgefahr, Augenschäden!). Diesen Nachteil vermeiden die elektrisch löschbaren PROMs. Sie verhalten sich (fast) so wie ein RAM, das nach dem Einschreiben der Information die Daten permanent hält, also auch nach dem Abschalten der Versorgungsspannung; dennoch läßt sich der Inhalt bei Bedarf löschen und verändert wieder eingeben. Genauso sieht die Idealvorstellung von einem Speicher aus, die heute nur deshalb noch nicht ganz erreicht ist, weil Lösch- und Programmierzeiten noch Sekunden bis Minuten in Anspruch nehmen, elektronisch gesehen also eine Ewigkeit dauern. Zum Teil brauchen diese elektrisch löschbaren PROMs geradezu exotische Versorgungsspannungen, um programmiert zu werden, was neben dem hohen Preis der gravierendste Nachteil ist.

7 Dokumentation

Chronologisch an der letzten Stelle und deshalb oft sträflich vernachlässigt steht die Dokumentation der funktionstüchtigen Programme. Darunter ist die Fixierung sämtlicher vom Programm ausgeführten Schritte zu verstehen, ergänzt durch erläuternden Kommentar. Es ist ein Phänomen beim Umgang mit der Software, daß man auch Programme, an deren Entwicklung man selbst lange Zeit zugebracht hat, schon nach zwei Wochen nicht mehr nachvollziehen kann. Es ist deshalb unerläßlich, die einzelnen Funktionen zumindest insoweit schriftlich zu erläutern, daß man für sich selbst eine fundierte Arbeitsunterlage besitzt. Optimal ist diese dann gestaltet, wenn aus den Unterlagen auch ein Fremder ersehen kann, wie das Programm im einzelnen aufgebaut ist und arbeitet.

Eine wesentliche Hilfe bei der Dokumentation ist ein Drucker, der in einfachen Ausführungsformen schon sehr preiswert zu haben ist. Damit läßt sich beispielsweise jederzeit der RAM-Inhalt bei der ROM-Simulation ausdrucken, um den jeweils aktuellen Entwicklungsstand festzuhalten (*Bild 16*). Durch die TTL-kompatiblen Anschlüsse und den unkritischen Aufbau (keine Kopfpositionierung) lassen sich derartige Einfach-Drucker problemlos ansteuern.

Zusammenfassend kann man sagen, daß auch für den Amateur ein eigener Mikrocomputer mit den dazugehörigen Peripheriegeräten kein unerfüllbarer Wunschtraum bleiben muß. Nachdem arbeitsfähige Kleinst-Computer bereits für weniger als 200,– DM angeboten werden, läßt sich ein solcher Aufbau als Grundlage für eigene Arbeiten heranziehen, die schrittweise bis hin zur Weiter- oder Neuentwicklung führen können.

Hinweis

Weder Redaktion noch Autor sind zeitlich dazu in der Lage, Entwürfe oder Modifikationen für Hard- und Software-Probleme anzufertigen.

Eine für den Anfänger konzipierte Einführungsreihe in die Mikrocomputer-Programmierung begann in Heft 9/1978 der Zeitschrift ELO; dort werden sämtliche Detailinformationen zum Aufbau und Betrieb eines eigenen Mikrocomputers gegeben.

Literatur

1 *Gößler, R.*: Entwicklungshilfsmittel für die Mikrocomputer-Programmierung. ELEKTRONIK 1977, H. 5, S. 36...43.

2 Internationale Organisation für Normungsfragen (ISO) und DIN-Norm 1355.

3 *Gößler, R.*: Einführung in die Mikrocomputer-Programmierung. ELEKTRONIK 1977, H. 9, S. 71...78.

4 ISP-8A/600 Single-Chip 8-bit N-Channel Microprocessor (SC/MPII). Datenblatt der Firma National Semiconductor, März 1977.

5 SC/MP Microprozessor Applications Handbook. Handbuch der Firma National Semiconductor, Februar 1977.

Hans-Georg Joepgen

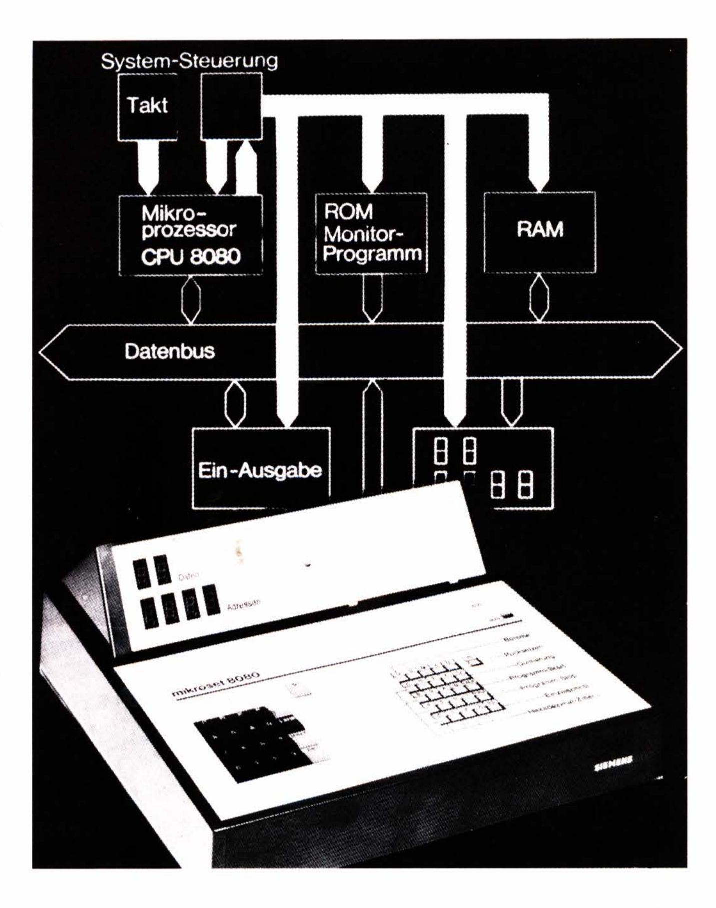
Herzstück eines Komplettsystems ohne Kompromisse:

Mikroset 8080 – ein „Lern- und Hobbycomputer"

Der billigste, wenngleich nicht der schnellste Weg zum Aufbau eines Hobbycomputer-Systems führt über Karteneinheiten, wie sie als Bausatz oder geprüfte Fertigplatinen angeboten werden. Dem Vorteil relativ geringer Anschaffungskosten stehen allerdings auch Nachteile gegenüber. So muß man für ein geeignetes Netzteil selbst sorgen, und kurzzeitige Spannungsspitzen oder Defekte in ungeeigneten Stromversorgungen haben schon so mancher CPU das Lebenslicht ausgeblasen. Darüber hinaus verzichten viele Kit- und Karten-Produzenten aus Kostengründen auf eine vollständige Decodierung der Adressen. Dadurch wird der Speicherraum, mit dem der Mikroprozessor von Haus aus umgehen könnte, erheblich reduziert. Die Folgen: Bei späteren Erweiterungen sind oft aufwendige Eingriffe in die Platine erforderlich. Diesen Nachteilen geht man aus dem Weg, wenn man sein System um den fix und fertig im Gehäuse gelieferten Rechner Mikroset 8080 (Siemens) aufbaut, den der Autor erprobte und kritisch bespricht.

Wenn Leute, die beruflich mit der drahtgebundenen Nachrichtentechnik zu tun haben, erstmals dem Mikroset begegnen, dann kommt ihnen der Rechner in aller Regel merkwürdig bekannt vor – ohne, daß sie auf Anhieb zu sagen vermögen, warum das so ist. Des Rätsels Lösung: Die Mikroset-Väter, Ingenieure der österreichischen Siemens-Gesellschaft, steckten den Computer in das Gehäuse einer weitverbreiteten Fernsprech-Tischvermittlung. Wo bei dieser Vermittlungseinrichtung allerdings Leuchtzeichen belegte und gerufene Anschlüsse melden, da zeigt beim Mikroset eine Sieben-Segment-Anzeigeeinheit Daten, Registernamen und Adressen an: alles in sedezimaler Maschinensprache, 0...9, dann A...F. Und wo die Vermittlungsstelle Wahl- und Schalttasten trägt, da hat der Mikroset einen Netz-Schiebeschalter, eine rote „Notbremse" in Form eines Zwangs-Reset-Knopfes sowie – neben seinen 16 Datentasten – neun Befehlsknöpfe, mit denen man während des Programmierens oder bei Testläufen allerlei nützliche Dienstleistungen des komfortablen Betriebssystems aufruft. Sechs dieser neun Tasten oder Kombinationen dieser Tasten kann man übrigens im Programm selbstgewählte Funktionen zuweisen. Erst nach Rückkehr ins fest eingespeicherte Betriebsprogramm bekommen sie ihre angestammten Bedeutungen zurück. Doch bleiben wir erst noch bei der Hardware. Zwei Fenstergruppen mit den Displays sind mit „Daten" und „Adressen" bezeichnet. Auch hier gilt die Beschriftung allein für den Fall, daß der Rechner im Betriebsprogramm auf Anweisungen wartet oder Aufträge erfüllt. Bei der Bearbeitung von Benutzer-Programmen hat man's in der Hand, Funktion und Bedeutung der Zeichen festzulegen.

Auf der Rückseite des Rechners: Anschlüsse für Netzspannung, für einen Kassettenrecorder, eine vierundsechzigpolige Federleiste als Zugang zur programmierbaren Schnittstelle SAB 8255, und: ein 96poliger Anschluß, der unbeschränkten Zugriff auf Adreßbus, Datenbus und Steuerbus erlaubt. Schon daran wird erkennbar: Der Mikroset präsentiert sich als eine Art Universal-Genie ohne Bauart-Kompromisse, das im Prinzip alles das kann, was Mikroprozessoren heute mit Fug und Recht zugemutet wird: Zusammenarbeit mit Druckern, Videoterminals und



Fernschreibmaschinen, mit externen Speicherblökken, mit Lochstreifengeräten; mit Sensoren, Motoren, Relais und Lämpchen: Was immer man ansteuern will (und mit einer TTL-kompatiblen Schnittstelle versieht), der Mikroset akzeptiert's als Partner.

Beim Autor bewährte sich der Rechner als Produzent einer Art von Elektronenmusik für eine Hörfunksendung. Die „Anpassungsschaltung" für ein Kofferradio, vor dessen Lautsprecher dann das Rundfunk-Mikrofon stand, war in diesem Fall übrigens ein schlichter 220-kΩ-Widerstand, der lediglich die Aufgabe hatte, den für den Tonband-Eingang eines Radios doch etwas brutalen TTL-Pegel auf bekömmliche Werte herabzusetzen.

1 Das Innenleben

Und wie sieht's im Bauch des Mikroset aus? Um eine schon etwas bejahrte CPU, die zum Betrieb dreierlei Versorgungsspannungen und als Assistenten die Hilfs-ICs System-Steuerbaustein und Taktgeber-Schaltung braucht, gruppiert sich ein mittelgroßes IC-Bergwerk (*Bild 1*): Alles nicht mehr ganz junge, aber bewährte und gestandene Kameraden wie der RAM-Baustein SAB 8111, der mit nur 256 x 4 bit bereits bis zum Kragen mit Weisheit gefüllt ist. Eine Technik, die reichlich Strom kostet, deshalb sind auch fünf integrierte Spannungsregler nötig.

Der Aufbau ist in verläßlicher kommerzieller „Strickart" ausgeführt, mit Abstandsbolzen, Kontermuttern, Steckverbindern. Eine Freude für jeden, der das Pappe- und Plastik-Innenleben vieler Geräte der heutigen Konsumelektronik nur mit Unbehagen zur Kenntnis nimmt.

Der verfügbare RAM-Bereich 0600-07FF des Mikroset wird vom Betriebssystem mitbenutzt, bei größeren Programmen besteht deshalb die Gefahr, daß der von der CPU angelegte Stapelspeicher ins Programm hineinwächst und Teile davon überschreibt. Erweiterung ist jedoch ohne Schwierigkeit möglich: Wie *Bild 2* erkennen läßt, sind bereits Fassungen für weitere RAM-Bausteine auf der CPU-Platine enthalten. Steckverbindungen für weitere Speicherkarten sind einschließlich mechanischer Halterungen ebenso vorgesehen wie die nötigen Reserven der eingebauten Netzteile und der Wärmeabfuhr. Zugriff auf umfangreichere Dateien kann unschwer über den Bus-Stecker erfolgen, allerdings müssen diese externen Speicherblöcke dann eine eigene Stromversorgung erhalten. Natürlich kann der Mikroset auch mit externen Festwertspeichern zusammenarbeiten, die beispielsweise einen BASIC-Interpreter enthalten, und sich mehrerer Floppy-Disk-Einheiten bedienen: Wie gesagt, die Architektur des Rechners enthält keinerlei bauartbedingte Einschränkungen, wenn man einmal davon absieht, daß die CPU 8080A nicht gerade einer der schnellsten Mikroprozessoren ist. Aber was soll's, kritische Echtzeitanwendungen mit hohem Datendurchsatz sind ohnehin nicht die Domäne der Hobbyelektronik. Für eine Mondlandungs-Simulation unter

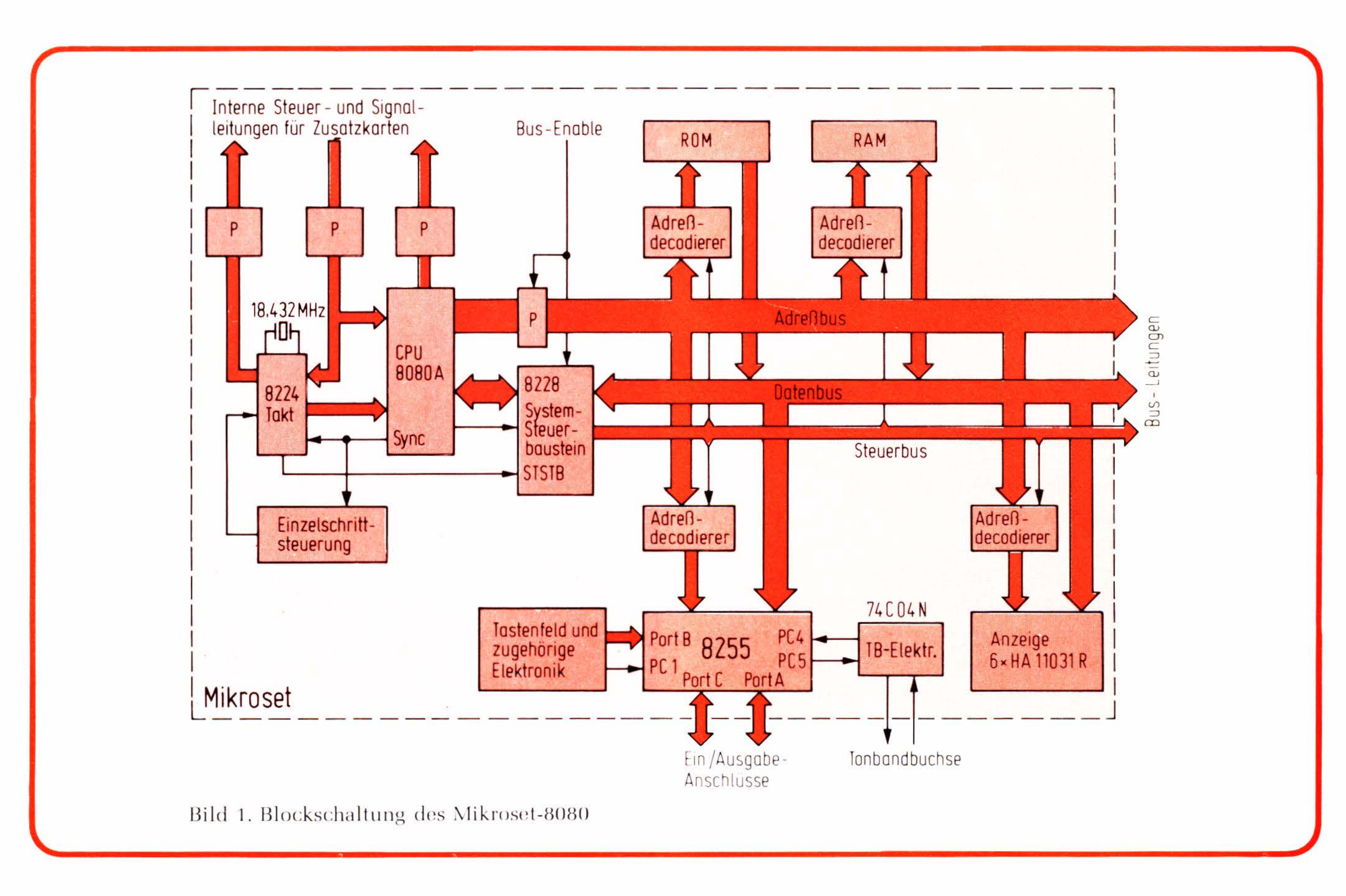


Bild 1. Blockschaltung des Mikroset-8080

Echtzeitbedingungen ist der Mikroset bei weitem schnell genug.

2 Das Betriebsprogramm

Man sieht, die Hardware läßt keine Wünsche offen, die man mit einiger Berechtigung an das Herzstück einer Hobbyanlage stellen darf – wie sieht es aber mit der Betriebssoftware aus? Im fest abgespeicherten Betriebsprogramm des Mikroset liegen seine Hauptvorzüge, denn im Unterschied zu mancherlei Konkurrenz-Monitoren sind die Siemens-Betriebsroutinen einschließlich ihrer Unterprogramme nahezu alle mit dem Maschinenbefehl C 9 abgeschlossen, was in Assemblersprache „Return" heißt. In Praxis bedeutet dies, daß von Benutzerprogrammen mit dem unbedingten oder einem bedingten Aufruf jederzeit eine Subroutine des Betriebsprogrammes in Marsch gesetzt werden kann, die dann abgearbeitet wird, bis nach dem Return-Befehl der Mikroset zurück ins Anwenderprogramm springt. Und da das Betriebssystem eine ganze Anzahl höchst nützlicher und relativ komplexer Handreichungen zur Verfügung stellt, kommt unversehens ein Hauch von Komfort in das ansonsten extrem mühselige Geschäft des Programmierens in Maschinensprache.

Im einzelnen verfügt das Betriebssystem über die Standard-Hardwarebefehle wie „Speicher laden", „Speicherinhalte und Registerinhalte anzeigen", „Benutzerprogramm abfahren" und „Reset", wenn sich der Rechner nach einem Programmierfehler im Kreise dreht und selbst keinen Ausweg mehr findet, es gehören dazu „Umladen der CPU-Register einschließlich Stapelzeiger und Bedingungsregister von Hand" und so fort. Darüber hinaus gibt es Einzelschritt-Abarbeitung und einen „Move" genannten, höchst nützlichen Tastenbefehl, der Datenblöcke in andere Speicherbereiche transportiert. Die „Display" genannte Möglichkeit, von einer gewählten Speicheradresse aus selbsttätig, im Sekundenrhythmus wechselnd und fortlaufend, den Speicherinhalt auf dem Displayfeld auszugeben, diese Möglichkeit hat, für den Autor jedenfalls, keinen praktischen Nährwert gezeigt und sich mehr als Spielerei entpuppt. Zum Notieren, Vergleichen oder gar Mitdenken ist eine Speicheranzeige in flexiblem Tempo, mit Wechsel nach Betätigung einer Quittungstaste, sinnvoller. Aber auch das kann der Mikroset.

3 Etwas Kritik

Nach soviel schmeichelhaften Feststellungen fragt sich der Leser sicherlich, ob es denn am Mikroset nichts auszusetzen gebe? Doch und ja, es gibt. Da ist einmal der Preis. Wenn man für etwas weniger als 3000 Mark beispielsweise im PET 2001 von Commodore einen mit 8 KByte ausgestatteten, mit Kassettenrecorder, Bildschirmterminal und IEC-Bus-Anschluß versehenen Computer bekommt, dann erscheinen runde 2200 DM einfach zu hoch. Bei Vergleichen ist

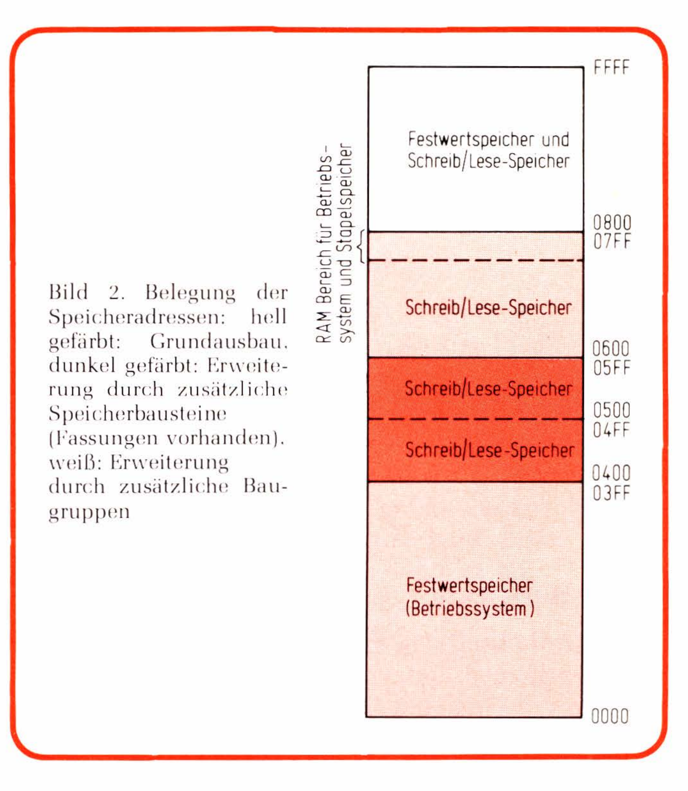


Bild 2. Belegung der Speicheradressen: hell gefärbt: Grundausbau, dunkel gefärbt: Erweiterung durch zusätzliche Speicherbausteine (Fassungen vorhanden), weiß: Erweiterung durch zusätzliche Baugruppen

allerdings einige Vorsicht geboten. Denn wer sich ein kleines oder auch größeres System auf 8080-Basis zusammenstricken will, und dieser Mikroprozessor ist trotz der Patina, die er unterdessen angesetzt hat, noch immer so etwas wie der KÄFER unter den Autos, für den ist der Mikroset die Entwicklungs- und später auch die Steuereinheit der Wahl.

Ein wenig bemängelt werden muß das mitgelieferte Papier. Wenn man den Anspruch der Wiener Mikroset-Väter ernst nehmen soll, ihr Kind sei auch ein ‚Lerncomputer', dann bedarf das mitgelieferte Dokumentationsmaterial dringend einer Erweiterung und didaktischen Aufbereitung. Von einem kleineren Monitor-Fehler, eine Diagnose-Subroutine verfälscht höchst heimtückisch ein Flag-Bit, von diesem Fehler hört man aus München, er sei unterdessen so gut wie behoben.

4 Für wen zu empfehlen?

Wem kann der Kauf des Mikroset-8080 empfohlen werden? Dem Hobby-Elektroniker dann, wenn er sich auf die CPU 8080 festgelegt hat; es gibt noch immer gute Gründe für eine solche Entscheidung.

Empfehlenswert ist dieses Gerät auch dann, wenn der didaktische Nutzen des Programmierens in Maschinensprache gefragt ist. Ein Elektronik-Anfänger begreift nicht mehr, was eigentlich genau im PET vorgeht, der Mikroset dagegen dürfte für jedermann verständlich sein, der jemals ein paar TTL-Gatter und eine integrierte ALU auf dem Experimentierbrett hatte. Was den Autor angeht, so lautet seine Entscheidung: Mikroset und Basic-Maschine; zur Zeit arbeitet er (mit viel Vergnügen) an einem Cross-Assembler für den Mikroset, der eines Tages auf dem PET laufen soll.

Der KIM-1 dürfte zur Zeit zweifellos zu den verbreitetsten Mikrocomputern auf dem Hobbymarkt gehören. Wir bringen daher in diesem Heft auch einiges für die Besitzer dieses etwa 800 DM „teuren" Systems, beschränken uns dabei aber auf die Programmierung im Maschinencode.

Herwig Feichtinger

KIM-1 – mehr als nur ein Spielzeug

1 KIM-Hardware

Der KIM-1 besitzt nicht nur ein sechsstelliges LED-Siebensegment-Display (normalerweise dient es zur Anzeige von Speicheradressen und Daten) und eine Sedezimal-Tastatur zur Eingabe in Maschinensprache, sondern auch ein Fernschreiber-Interface (ASCII, 20 mA) und eine Schaltung, die das Aufzeichnen von Programmen bzw. Datenblöcken auf einem gewöhnlichen Mono-Kassettenrecorder erlaubt.

Bild 1 zeigt die Speicherbereichs-Verteilung des KIM-1. Die „Zero Page" (Seite Null) hat bei dem verwendeten Mikroprozessor 6502 von MOS Technology eine besondere Bedeutung; Adressen in diesem Speicherbereich können mit einer Art von Kurzform-Adressierung angesprochen werden, wobei die führenden Nullen weggelassen werden. Dadurch spart man bei dem jeweiligen Befehl ein Byte. Ein Teil der „Zero Page" wird bereits von dem in einem ROM gespeicherten Monitor-Programm benötigt, das für die Anzeige, für die Tastenfeld-Decodierung und die Magnetband-Aufzeichnung dient.

Der ROM-Bereich ist auf die beiden Chips 6530-002 und 6530-003 verteilt, die auch I/O-Ports und insgesamt zwei Zeitgeber (Timer) enthalten. Ferner besitzen sie je einen 64 x 8-bit-RAM-Speicher.

Zur Speichererweiterung sind verschiedene Zusatzplatinen lieferbar, ebenso für den Anschluß an den in den USA recht verbreiteten S-100-Bus und zur Darstellung von ASCII-Zeichen auf einem Fernsehgerät.

2 Die CPU 6502

Der Mikroprozessor 6502 ist hardwaremäßig sehr mit dem 6800 verwandt; beide sind – bis auf eine Ausnahme – sogar stiftkompatibel. Bei der Software sieht es allerdings etwas anders aus. Der 6502 besitzt einen Akkumulator und zwei Indexregister (X und Y); beim 6800 ist es genau umgekehrt.

Der 6502 bietet eine große Anzahl von Adressierungsmöglichkeiten: „Immediate", „Absolute", „Zero Page" (s. o.), „Accu", „(Indirect, X)", „(Indirect), Y", „Zero Page, X", „Absolute, X", „Absolute, Y", „Relative" (bei bedingten Sprüngen), „Indirect" (bei unbedingten Sprüngen) und „Zero Page, Y".

Die bedingten Sprünge (Branch-Befehle) erlauben die Adressierung von max. 127 bzw. 128 Bytes vor- bzw. rückwärts und benötigen nur zwei Bytes (8080: drei Bytes).

Von den bei 8 bit Befehlslänge möglichen 256 Kombinationen sind knapp 150 als Befehle ausgenützt. Das 8-bit-Status-Register (S) enthält Flags für Negative, Overflow, Break (das ist eine Art Software-Interrupt), Dezimal/Sedezimal (in beiden Systemen kann der 6502 addieren und subtrahieren), Interrupt-Sperrung, Null, Übertrag.

Eine besondere Eigenschaft des 6502 ist auch der Speicherzugriff im „Pipelining"-Verfahren, der dafür sorgt, daß jede Taktphase ausgenützt wird. Er ist einer der Gründe, warum der BASIC-Interpretierer für den 6502 einer der schnellsten überhaupt ist. Doch davon nur am Rande; wir wollen uns hier ja auf die Verwendung der „rohen" KIM-Platine beschränken.

3 Nützliche Monitor-Unterprogramme

Wie schon erwähnt, dient das im KIM als ROM gespeicherte Monitorprogramm unter anderem dazu, die Tasteneingabe zu ermöglichen und auf dem sechsstelligen Display in sedezimaler Form vierstellige

Erweiterung bis 64 K	FFFF 2000	Timer und I/O	173F 1700
ROM 6530-002	1 FFF 1C000	Frei für Erweiterungen, z. B. RAM	16FF 0400
ROM 6530-003	1BFF 1800	Freies RAM	03FF 0200
Monitor-RAM	17FF 17EC	Stack-Bereich	01FF 0100
Freies RAM	17EB 1780	Monitor-RAM	00FF 00EF
Monitor-I/O	177F 1740	Freies RAM (Zero Page)	00EE 0000

Bild 1. Speicherbereichsverteilung beim KIM-1

Adressen und zweistellige Daten darzustellen. Eine ganze Reihe von Monitor-Unterprogrammen kann aber auch in Anwenderprogrammen eine nützliche Verwendung finden; *Tabelle 1* zeigt die interessantesten.

Auch für die Art und Weise, aus einem Anwenderprogramm durch einen Sprung ins Monitorprogramm zurückzukehren, gibt es beim KIM-1 einige Möglichkeiten, wie *Tabelle 2* zeigt. Diese Tatsache erweist sich besonders während der Programm-Entwicklung manchmal als sehr nützlich.

Und *Tabelle 3* zeigt schließlich, wie einige Monitor-Unterprogramme benutzt werden, um ein Speicher-Testprogramm zu realisieren. Es prüft, ob sich in allen Zellen die Daten 00 und FF speichern lassen; damit werden jeweils alle 8 bit auf „Low" und „High" getestet. Der Test beginnt bei der Adresse, deren niederwertiges Byte in der Zelle 0000 und deren höherwertiges Byte in der Zelle 0001 steht und hält bei der Adresse an, bei der er zum erstenmal versagt. Natürlich ist es nicht möglich, die „Zero Page" damit zu testen, da dort ja das Testprogramm selbst steht. Das Programm sei aber jedem empfohlen, der sich gerade einen KIM-1 oder eine zusätzliche RAM-Karte erwarb, um (vor Ablauf der Garantiefrist!) die richtige Funktion aller Speicherzellen zu prüfen. Das Monitor-Unterprogramm SCAND erlaubt es dabei, den Ablauf zu verfolgen; es sorgt dafür, daß die gerade getestete Adresse (und die Daten FF) in der Anzeige zu sehen sind. Der Ablauf läßt sich erheblich beschleunigen, wenn auf diese Anzeige verzichtet wird, weil der Unterprogramm-Aufruf SCAND einige Millisekunden in Anspruch nimmt. Die drei Bytes 20 19 1F (Adresse 001B) wären dann durch EA EA EA (No Operation) zu ersetzen. Dies ist besonders bei sehr umfangreichen Speicherblöcken sinnvoll.

4 Magnetband-Aufzeichnung

Das im KIM-1 verwendete PLL-System für die Wiedergewinnung der Digitalsignale aus Frequenz-Umtastsignalen vom Kassettenrecorder ist gegenüber Phasenschwankungen und Amplitudenänderungen bis zu etwa 10 dB recht unempfindlich. Schwierigkeiten können aber auftreten, wenn

- vor dem Aufnahme- bzw. Wiedergabestart nicht die Daten 00 in die Zelle 00F1 geladen werden, um die sedezimale Arbeitsweise sicherzustellen;
- die (nur für die Wiedergabe benötigte) 12-V-Spannung mehr als etwa 100 mV Brumm aufweist;

Tabelle 1. Nützliche Monitor-Programme

Name	Adresse	Verlorene Register	Wirkung
SCANS	1F1F	A, X, Y	Zeigt den Inhalt der Zellen FB, FA, F9 im Display an. Wenn eine Taste gedrückt ist, wird das Zero-Flag gleich Null gesetzt.
SCAND	1F19	A, X, Y	Zeigt die durch FB, FA spezifizierte Adresse und die dort gespeicherten Daten an. Sonst wie SCANS.
GETKEY	1F6A	A, X, Y	Akku enthält die der gedrückten Taste entsprechenden Daten. A ≧ 15 (hex.): keine Taste gedrückt.
INCPT	1F63	–	Erhöht die Zelle FA um 1. Wenn dabei ein Übertrag auftritt, wird auch FB um 1 erhöht.
GETCH	1E5A	A, Y	Lädt den Akku mit einem TTY-Zeichen.
PRTBYT	1E3B	Y	Druckt den Akkuinhalt als zwei Hex-Ziffern auf dem Fernschreiber.
PRTPNT	1E1E	A, Y	Druckt den Inhalt der Zellen FB und FA auf dem Fernschreiber.
OUTCH	1EA0	A, Y	Druckt den Akku-Inhalt als ASCII-Zeichen auf dem Fernschreiber.
OUTSP	1E9E	A, Y	Druckt einen Leerraum auf dem Fernschreiber.

Tabelle 2. Rückkehr ins Monitor-Programm

Name	Adresse	Gerettete Register	Angezeigte Adresse	Bemerkungen
SAVE	1C00	A,X,Y,S	aktuelle A.	Meist als NMI-Vektor; günstig auch als IRQ-vektor für BRK-Befehl
RST	1C22	–	Programm-startadr.	Stack-Pointer wird rückgesetzt, I/O-Ports werden als Eing. geschaltet.
START	1C4F	–	Programm-startadr.	Keine Wirkung auf den Status.
–	1925	–	0000	Praktisch bei Programmen die die Zelle 0000 als Argument benutzen.
LOAD9	1929	–	FFFF	Normalerweise als Fehler-Anzeige.

Tabelle 3. Speicher-Testprogramm

Adresse	Mnemonisch	Sedez.-Code
0002	LDA Z,00	A5 00
0004	STA Z,FA	85 FA
0006	LDA Z,01	A5 01
0008	STA Z,FB	85 FB
000A	LDY I,00	A0 00
000C	TYA	98
000D	STA(FA), Y	91 FA
000F	CMP(FA),Y	D1 FA
0011	BNE 11	D0 11
0013	LDA I,FF	A9 FF
0015	STA(FA),Y	91 FA
0017	CMP(FA),Y	D1 FA
0019	BNE 09	D0 09
001B	JSR SCAND	20 19 1F
001E	JSR INCPT	20 63 1F
0021	JMP 000A	4C 0A 00
0024	JMP 1C22	4C 22 1C

– das Tonbandgerät schon bei etwa 3,6 kHz einen starken Amplitudenabfall aufweist;
– wenn die Bandkassette einen hohen „Drop-Out"-Anteil aufweist;
– der Wiedergabeverstärker bei abgeschaltetem Lautsprecher, d. h. hochohmiger Last, zum Schwingen neigt.

Die Störanfälligkeit gegenüber „Drop-Outs" (wenn sie etwa 15 dB nicht übersteigen) und niederfrequenten Störgeräuschen, wie etwa 50-Hz- oder 100-Hz-Brumm, läßt sich erheblich vermindern, wenn auf der KIM-1-Platine der Koppelkondensator C 6 (0,22 µF) auf etwa 4,7...10 nF verringert wird. Zur Überwachung des Wiedergabevorganges ist es auch praktisch, den Kassettenrecorder so umzubauen, daß der eingebaute Lautsprecher nicht beim Einstecken der Nf-Verbindungsleitung abgeschaltet wird. Dazu ist meist lediglich die Verdrahtung an der Ohrhörer-Buchse zu ändern. Damit es bei dem für den KIM-1 günstigsten Nf-Pegel nicht zu laut wird, kann man in Serie zum Lautsprecher einen Vorwiderstand von z. B. 47 Ω schalten.

Manche Bandfabrikate besitzen leider einen katastrophal hohen „Drop-Out"-Anteil. Mit gewissen Einschränkungen läßt sich aber sagen: Je teurer, desto besser. Sollte sich aber das Drop-Out-Problem nicht beseitigen lassen, so ist entweder der Tonkopf verschmutzt oder die in der Kassette enthaltene Band-Andruck-Feder nach innen verbogen.

5 Siebensegment-Alphabet

Wer keinen Fernschreiber oder kein Datensichtgerät besitzt (ersterer ist laut, zweiteres teuer), wird vielleicht die Möglichkeit begrüßen, das Siebensegment-Display des KIM-1 zur Anzeige eines Pseudo-Alphabets zu verwenden. Zugegeben, die Ziffern und Zeichen in *Bild 2* sehen etwas seltsam aus, aber was will man von sieben Segmenten schon verlangen...

Das in *Tabelle 4* aufgelistete Programm ermöglicht es, die Inhalte der Zellen 0006...0001 als ASCII-Zeichen zu interpretieren und als Zeichen so darzustellen, wie das aus Bild 2 ersichtlich ist. Dabei benutzt es eine Tabelle, die bei der Adresse 0300 beginnt; die Zeichenform läßt sich durch Ändern der Tabellendaten natürlich beliebig abwandeln, da im KIM-1 die Codeumwandlung nicht durch festverdrahtete Decodierer, sondern allein durch die Software geschieht. Zeichen, die nicht in Tabelle 4 enthalten sind, werden lediglich als Leerstelle interpretiert. Sicher gewöhnt man sich recht schnell an die Darstellungsform, und wenn auch Groß- und Kleinbuchstaben gemischt dargestellt werden müssen, so ist das Siebensegment-Alphabet doch die preiswerteste Möglichkeit der Darstellung von ASCII-Zeichen.

Späteren Erweiterungen sind natürlich kaum Grenzen gesetzt; Drucker, Sichtgerät, RAM-Karten haben nur eine Voraussetzung: ein gefülltes Sparschweinchen.

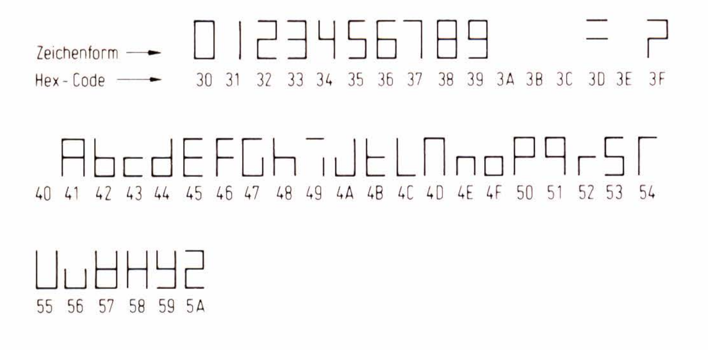


Bild 2. Etwas ungewohnt sieht das Siebensegment-Alphabet schon aus, aber immerhin enthält es alle Buchstaben und einige Zeichen

Tabelle 4. Siebensegment-Pseudoalphabet

Adresse	Mnemonisch	Sedez.-Code		
0200	LDA I,7F	A9	7F	
0202	STA 1741	8D	41	17
0205	LDX I,09	A2	09	
0207	LDY I,06	A0	06	
0209	LDA 0000,Y	B9	00	00
020C	STY Z,FC	84	FC	
020E	TAY	A8		
020F	CPY I,30	C0	30	
0211	BCC 04	90	04	
0213	CPY I,5B	C0	5B	
0215	BCC 02	90	02	
0217	LDY I,2F	A0	2F	
0219	LDA 02D1,Y	B9	D1	02
021C	LDY I,00	A0	00	
021E	STY 1740	8C	40	17
0221	STX 1742	8E	42	17
0224	STA 1740	8D	40	17
0227	LDY I,7F	A0	7F	
0229	DEY	88		
022A	BNE FD	D0	FD	
022C	INX	E8		
022D	INX	E8		
022E	LDY Z,FC	A4	FC	
0230	DEY	88		
0231	BNE D6	D0	D6	
0233	STX 1742	8E	42	17
0236	LDA I,00	A9	00	
0238	STA 1741	8D	41	17
023B	JMP 0200	4C	00	02
Umwandlungs-Tabelle: (Leerstelle: –)				
0300	–,0,1	80	BF	86
0303	2,3,4	DB	CF	E6
0306	5,6,7	E6	FD	87
0309	8,9,–	FF	EF	80
030C	–,–,=	80	80	C1
030F	–,?,–	80	D3	80
0312	A,B,C	F7	FC	D8
0315	D,E,F	DE	F9	F1
0318	G,H,I	BD	84	85
031B	J,K,L	9E	F8	B8
031E	M,N,O	B7	D4	DC
0321	P,Q,R	F3	E7	DO
0324	S,T,U	ED	B1	BE
0327	V,W,X	9C	FE	F6
032A	Y,Z	EE	DB	

Hans-Georg Joepgen

Ein Mini-Entwicklungssystem zu kleinem Preis

Die von verschiedenen Firmen angebotenen Mikrocomputer-Entwicklungssysteme, in aller Regel ausgestattet mit Bildschirmausgabe, Floppy-Disk-Laufwerk und Emulator, bieten reichhaltige Unterstützung und hohen Komfort beim Entwurf und beim Austesten von Mikroprozessor-Programmen und µP-Hardware-Konfigurationen. Wegen des noch immer sehr hohen Preises dieser Entwicklungssysteme ist ihr Einsatz jedoch nur dann zu rechtfertigen, wenn ein hoher Ausnutzungsgrad zu erreichen ist. In Firmen und Instituten, die nur gelegentlich vor der Notwendigkeit stehen, ein Mikrocomputer-System nach Maß zu entwickeln, würde man gern ein paar Ingenieurstunden mehr in Kauf nehmen, wenn sich dadurch die hohen Anfangskosten senken ließen. Hier führt ein gangbarer Weg über die Benutzung von „Prototyp-Kartensystemen"; für den Hobby-Elektroniker, für den in aller Regel die Anschaffung eines großen Entwicklungssystems ohnehin nicht in Betracht kommt, ist ein solcher „Kartencomputer" unterdessen gar zur Standard-Lösung geworden. Einen dieser Karten-Bausätze, der über ein besonders günstiges Preis/Leistungs-Verhältnis verfügt, den „System Design Kit SDK 85" von der Firma Intel, erprobte der Autor dieses Beitrages kritisch.

Hersteller autarker Einkarten-Mikrocomputer können beim Entwurf ihres Produktes zwei verschiedene Wege gehen. Entweder sie streben einen kleinen Endpreis an und verzichten auf vollständige Decodierung des Adreß-Busses und auf die Herausführung gepufferter Bus- und Steuersignale, dann kommt der Kunde in Schwierigkeiten, wenn er sein System vergrößern möchte. Oder aber, Möglichkeit Nummer zwei für den Produzenten, er entscheidet sich für eine kompromißlose, sprich voll decodierte und unbeschränkt gepufferte Schaltung, dann spürt's der Kunde am Preis – und der Hersteller am Umsatz. Intel hat mit seinem Bausatz SDK-85 versucht, zwei Fliegen mit einer Klappe zu schlagen. Der Kit kommt nämlich mit einer Platine ins Haus, die dem Käufer die Wahl läßt, ob und in welchen Stufen er erweitern will. Durch Einlöten einiger weiterer ICs und Pfostenstecker, durch Änderung weniger Drahtbrücken ist die Karte nämlich in wenigen Minuten vom Komplett-Einfachsystem in die Zentraleinheit eines Computers verwandelt, der 64 KByte und einige hundert Ein/Ausgabe-Leitungen zu handhaben weiß.

Wenn man gar drei, vier Ausgabeleitungen zum Setzen von sogenannten „Page-Flipflops" benutzt, die ihrerseits wieder externe Speicherblöcke umschalten, dann kann man mit dem SDK-85 in den MByte-Bereich vorstoßen – sofern das nötige Kleingeld vorhanden ist. Dieser hohe Grad von Flexibilität und die Vorzüge des SDK-85-Herzstücks, der CPU 8085, waren es, was den Autor bewog, den Bausatz aufzubauen und zu erproben.

Der Kit kommt mit einer stellenweise vorzüglich gemachten Dokumentation ins Haus. Schritt für Schritt wird in knappem, präzisem Englisch erklärt, in welcher Reihenfolge beim Zusammenbau zu verfahren ist. Checklisten, die abgehakt werden sollen, dienen als Kontrolle darüber, daß auch nichts vergessen wurde. Ausführlich wird erläutert, wie zu löten ist, daß die Augen beim Abknipsen überstehender Anschlußstifte geschützt werden müssen – und so fort.

Eine Anleitung, die sich an den Anfänger wendet: Aber hier ergibt sich ein Widerspruch. Die große Zahl eng beieinander liegender Lötstellen ist, darüber kann kein Zweifel sein, durch Leute ohne Erfahrung im Bestücken und Verlöten dichtgepackter Karten ohne unbeabsichtigte Brücken nicht zu meistern. Die Qualität der Leiterbahnauflage allerdings, und das ist Intel in diesem Punkt zugute zu halten, die Qualität der Leiterbahnbefestigung läßt durchaus einige Reparaturversuche zu, ehe es zum Ablösen des Leitmetalls kommt.

Zahlreiche Skizzen und Fotos in dieser vorzüglich gemachten Anleitung, Tips zum Verfahren („Räumen Sie vorher Ihren Arbeitstisch ab. Sorgen Sie dafür, daß Sie ungestört arbeiten können. Gehen Sie nicht ermüdet oder nervös ans Werk!") verdienen uneingeschränkten Beifall. Die vom Handbuch veranschlagte Arbeitszeit zum Zusammenbau, drei bis fünf Stunden („depending upon your skill – Abhängig von Ihrer Geschicklichkeit"), hat den Autor in tiefe Zweifel darüber gestürzt, ob er sich überhaupt noch zur Gilde der Hobby-Elektroniker rechnen darf, wiewohl er schon seit 21 Jahren den Lötkolben schwingt: Bei ihm dauerte es nämlich acht Stunden (Kaffeepause bereits abgezogen), ehe er sein „Es ist vollbracht!" sprechen konnte und in den LED-Displays die ersten Segmente aufblitzten.

Nochmal zwei Stunden heftigen Oszilloskopierens kamen dazu, bis klar war, warum zwei Segmente partout nicht aufhören wollten, zu glimmen, obgleich die CPU ein energisches „Licht aus!" befahl. Des Rätsels

Lösung: kein Lötfehler, kein zu heiß gewordener Transistor, keine durchgeschlagene Gate-Isolation, sondern offenbar eine Konstruktions-Eigenheit von optimistischen Konstrukteuren mit anscheinend unbegrenztem Vertrauen in die Datenblätter der eigenen Firma. Zwei zusätzlich eingelötete Basis-Ableitwiderstände, parallel zu den Basis-Emitter-Strecken von Treibertransistoren, brachten die Welt wieder in Ordnung.

Dann also liefen die ersten Programme, die das Handbuch zum Einschalttest vorschlägt. Reaktionszeitmesser, Zähler und dergleichen Spielchen mehr. Alles einwandfrei, fast nirgendwo ein Grund zur Klage. Fast kein Grund, weil das Tastenfeld auf der Platine bei jedem Tastendruck verrät, daß bei einem 300-Dollar-Computersystem gespart werden muß, um einen solchen Preis möglich zu machen. Verwöhnt durch den Siemens-Mikroset mit seinen präzis schaltenden Tasten und ihrem eindeutigen Druckpunkt, mochte sich der Autor nur sehr zögernd mit der Intel-Tastatur befreunden.

Unterdessen gibt es ja eine größere Zahl von Kartensystemen auf 8080-Basis. Herzstück des Bausatzes SDK-85 jedoch ist, der Name sagt's, der 8080-Nachfolger 8085. Die Vorzüge dieser CPU gegenüber dem Vorläufertyp kommen im besprochenen Kit voll zum Tragen: nur eine Betriebsspannung statt deren drei, deutlich höhere Arbeitsgeschwindigkeit bei gewissen Operationen, problemloser Umgang mit maskierbaren Interrupts, zusätzliche Restart-Möglichkeiten, erweiterter Befehlsvorrat, Schnittstelle für serielle Datenein- und -ausgabe bereits in der CPU integriert, ebenso wie die komplette Systemsteuerung und Takterzeugung. Alles durchaus angenehme Überraschungen für jemanden, der bisher allein mit dem guten alten 8080 zu tun hatte.

Das gleiche läßt sich über die Speicherbausteine der Grundausstattung sagen. Der RAM-IC, Typ 8155, bietet außer 2 Kbit Speicherraum noch zwei Ein/Ausgangs-Ports zu je 8 bit und einen mit 6 bit, ein jeder dieser Kanäle zum Datenverkehr mit der Außenwelt läßt sich von der CPU dazu veranlassen, höchst trickreiche Betriebsarten zu zeigen. Weiter enthält der 8155 einen 14-bit-Binärzähler, der Ereignisse aufsummieren, Frequenzen messen oder Zeitgeber spielen kann. Und all das, ohne die CPU zu bemühen, bei der sich der Zähler erst dann meldet, wenn er seine zugewiesene Aufgabe erfüllt hat.

Das komfortable Betriebssystem steckt in einem 16 384-bit-ROM mit Namen 8355. Damit sich dieser Speicherbaustein im Kreise seiner hochkomplexen Kameraden RAM und CPU nicht zu dürftig vorkomme, versahen ihn die Intel-Ingenieure noch mit zwei Ein/Ausgangs-Ports zu je 8 bit, von denen ein jedes als Eingang oder Ausgang zu programmieren ist.

Der Tausendsassa Nummer eins auf der SDK-85-Platine ist jedoch der Baustein 8279-5, Amtsbezeichnung „Keyboard and Display Controller". Dieser Tastenfeld- und Leuchtdioden-Spezialist ist ein kleines Mikrocomputer-Subsystem mit eigenen Speichern, das der CPU eine ganze Menge lästiger Kleinarbeit abnimmt, wie Tastenkontakte abfragen und das Display auf dem laufenden halten. All das in vielerlei Betriebsart-Varianten – wie und warum, mit welchen Vorzügen, dies zu schildern würde Seiten beanspruchen. Dazu nur soviel: Mit seiner Hilfe können die Leuchtdioden nicht nur die Ziffern 0...9 sowie die Sedezimalzahlen A...F melden, sondern auch so bedeutsame Worte wie „Hilfe" oder „Esel" aufleuchten lassen – und jedes Kurzwort aus Buchstaben, die sich aus dem Achter-Muster der klassischen Siebensegment-Anzeigeeinheit zusammensetzen lassen.

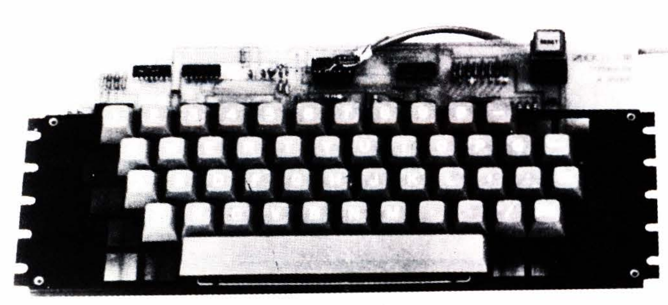
Schlußurteil des Autors über den Bausatz SDK-85 von Intel: Viel Leistung für relativ wenig Geld, aber trotz vorzüglicher Dokumentation für den Mikrocomputer-Anfänger nur dann geeignet, wenn er beim Löten über eine sichere Hand verfügt und den einfacheren 8080 zuvor bereits kapiert hat. Der komplexere 8085 und seine intelligenten Assistenten 8155, 8355 und 8279-5 strapazieren die Lernbereitschaft ihres Besitzers so schamlos, daß im Hirnkastel kaum noch Verarbeitungskapazität zum Erlernen etwa der Assemblersprache übrigbleibt. Reiten lernt man nicht auf Rennpferden.

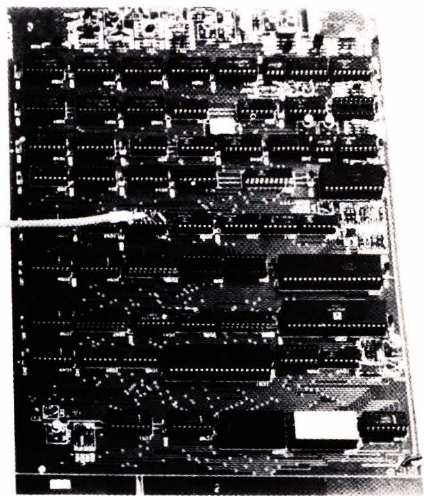
Technische Daten der Grundausstattung

Steuereinheit (CPU):	MCS 8085
Zykluszeit:	1,3 µs
ROM:	2 KByte
RAM:	256 Byte
E/A-Ports:	Parallel 38 Leitungen, TTL Seriell direkt über CPU, TTL
Baudrate für externes Terminal:	110,20-mA-Stromschleife
Interrupts:	Drei Ebenen: RST 7,5 vom Monitor belegt, RST 6,5 und INTR verfügbar (TTL)
Direkter Speicherzugriff (DMA):	Durch Umlöten einer Drahtbrücke herstellbar; TTL
Versorgung:	5 V ±5 %; 1,3 A (bei Teletype-Betrieb zusätzlich erforderlich: 10 V ±10 %, 300 mA)
Monitorbefehle bei Betrieb mit Terminal (Auszug):	
Reset:	Zwangsrückkehr ins Betriebsprogramm
Substitute Memory:	Speicher laden oder Inhalt ändern
Examine Register:	Registerinhalte anzeigen und ändern
Go-Execute:	Steuerung geht an Benutzerprogramm
Single Step:	Einzelschritt-Betrieb zur Fehlersuche
Next:	Fortsetzung des Programms
Vector Interrupt:	Zwangssprung nach RAM-Adresse 20D4
Bei Konsolenbetrieb zusätzlich möglich:	Transport geschlossener Speicherbereiche; Listing.

(Empfehlenswert: Gleich von Anfang an ein Zusatz-RAM 8155 mitbestellen und am vorgesehenen Platz einlöten)

Preiswerter Einkartencomputer mit Video-Interface





NASCOM 1 ist ein kompletter Einkartencomputer mit Video-Interface und externer alphanumerischer Tastatur. Das System benötigt zur Inbetriebnahme lediglich einen handelsüblichen Heimfernseher und eine Stromversorgung (+5 V, +12 V, −5 V). Mit Hilfe des Monitorprogramms kann sofort in Maschinencode programmiert werden. Bei der Konstruktion des Systems wurde darauf Wert gelegt, möglichst viele Hardware-Eigenschaften zu integrieren, die für komfortables und universelles Arbeiten erforderlich sind. Aus diesen Gründen wurde statt der üblichen Taschenrechnertastatur mit 7-Segment-Anzeigeeinheit eine Bildschirmdarstellung mit kompletter Tastatur verwendet. Trotzdem konnte ein Preis von unter 1000 DM erreicht werden.

1 Hardware

Das System besteht aus einer etwa 28 x 20 cm² großen Leiterplatte, auf der sich fünf LSI-Bausteine, sechzehn 1-Kbit-RAMs und 32 Low-Power-Schottky-TTL-Bausteine sowie diverse diskrete Bauelemente befinden. Alle ICs sitzen auf Sockeln. Für Erweiterungen ist der gesamte Bus an einem Direktsteckverbinder herausgeführt.

Als Parallelschnittstelle stehen dem Anwender ein kompletter Z-80-PIO mit zwei E/A-Ports an zwei 16poligen IC-Sockeln zur Verfügung. Die seriellen Schnittstellen werden von einem UART (*Universal Asynchronous Receiver/Transmitter* = Asynchroner Sender/Empfänger-Baustein) IM 6402 angesteuert. Für den Takt stehen drei Möglichkeiten zur Verfügung:

- 3,9 kHz für eingebautes Kassetten-Interface (etwa 250 Bd) von der Teilerkette abgeleitet;
- Oszillator mit NE 555, abstimmbar auf 110 Bd für z. B. TTY;
- externer Takt, das Übertragungsformat ist acht Datenbits ohne Parität mit zwei Stoppbits, beides kann hardwaremäßig geändert werden.

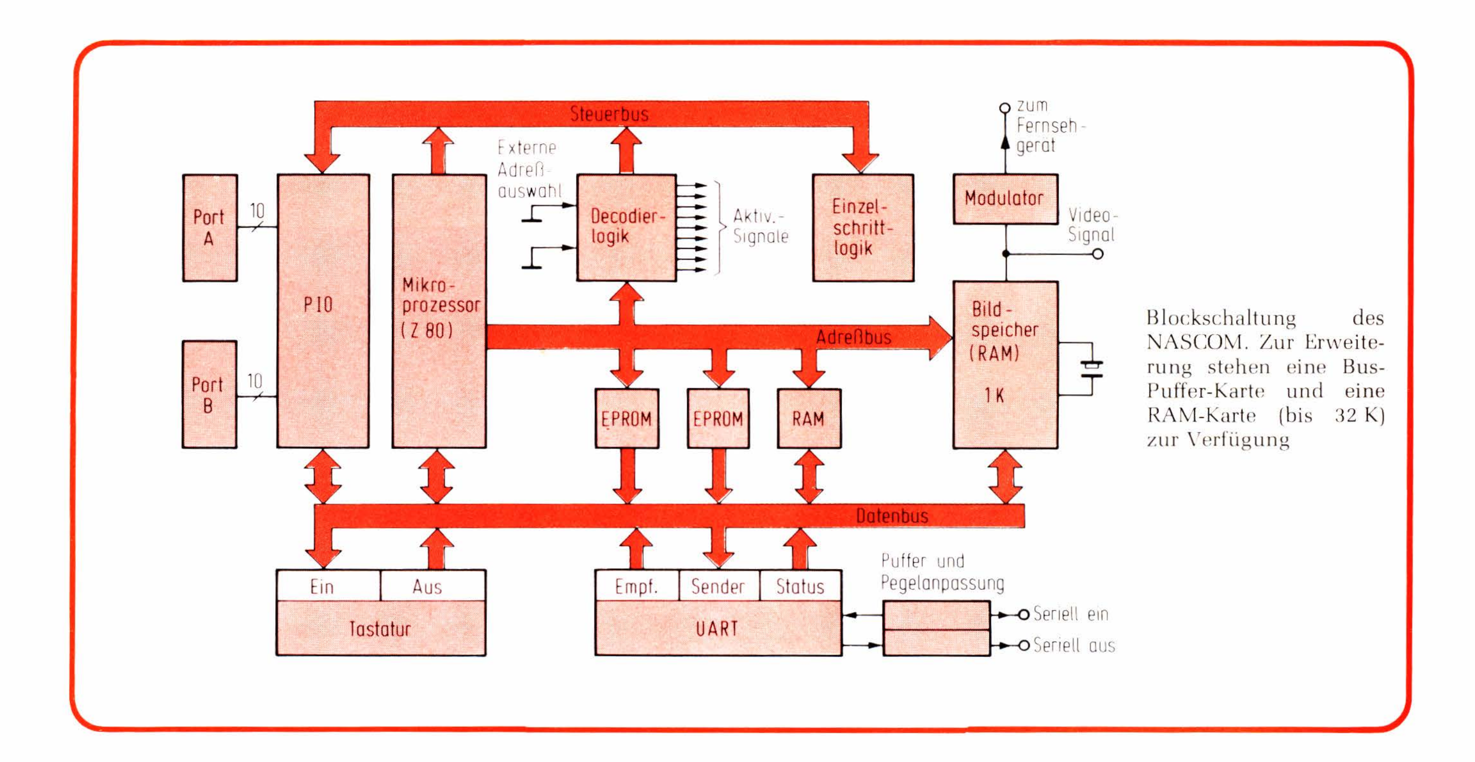
Als Schnittstelle stehen zur Verfügung: RS 232/V 24 und 20-mA-Stromschleife für TTY, beide sind an einem 16poligen IC-Sockel verfügbar. Das eingebaute Kassetten-Interface arbeitet mit 1,95-kHz-Frequenzbündeln.

Das Anwender-RAM ist mit acht statischen 1-Kbit-RAMs aufgebaut. Es ist von C00H...FFFH adressiert. Für festgespeicherte Programme sind zwei Sockel für EPROMs (2708) verfügbar.

Die Tastatur besteht aus 48 Tasten. Aus Kostengründen wurde auf einen Codierer-Baustein verzichtet und stattdessen eine auf minimalen Hardware-Aufwand reduzierte Matrixsteuerlogik verwendet. Sie besteht aus Zählern und Decodierern, die per Software getaktet und abgefragt werden. Die Steuersignale werden über ein 6-bit-Latch, das als Port 0 adressiert ist, über ein 16poliges Kabel zur Tastatur übertragen. Die Eingänge werden über einen 8-bit-Puffer (ebenfalls Port 0) abgefragt.

Das Video-Interface arbeitet mit einem 1-KByte-RAM, dessen Inhalt als ASCII-Zeichen interpretiert und als 7 x 9-Matrizen auf den Bildschirm gebracht wird.

Der Zeichengenerator enthält Groß- und Kleinbuchstaben – insgesamt 128 Zeichen. Über Multiplexer sind die Adreßleitungen der RAMs einerseits mit einer aus Zählern und Teilern bestehenden Adressier-



Blockschaltung des NASCOM. Zur Erweiterung stehen eine Bus-Puffer-Karte und eine RAM-Karte (bis 32 K) zur Verfügung

logik, andererseits mit dem CPU-Adreßbus verbunden. Die Datenein- und -ausgänge sind ebenfalls über Puffer mit dem CPU-Datenbus verbunden. Der Prozessor hat damit vollen Zugriff auf das Video-RAM und übernimmt alle Zeichenbewegungen des Cursors, „back space", „Aufrollen" und Löschen. Da die CPU Priorität hat, entstehen keine Zugriffsprobleme. Das Bildschirmformat ist durch die Hardware auf 16 Zeilen zu 48 Zeichen festgelegt.

2 Software

Das NASBUG-Monitorprogramm ist in zwei Blöcke aufgeteilt. Der eine Teil enthält die Unterprogramme für folgende Ein/Ausgaben:

Tastatur: Diese Routine fragt die Tastatur ab, bestimmt die gedrückte Taste, entprellt sie und wandelt den Wert in ein ASCII-Zeichen um.

Video-RAM: Diese Routine schreibt ein Zeichen in die nächstmögliche Position, setzt den Cursor und führt Line-Feed- und Carriage-Return-Kommandos aus.

Seriell ein: Liest ein Zeichen vom UART.

Seriell aus: Schreibt ein Zeichen in den UART.

Der zweite Teil umfaßt die Kommandos des Monitorprogramms. Sie sind für die komfortable Programmierung in Maschinensprache ausgelegt.

T: Dieses Kommando gefolgt von Start- und Endadresse listet die Daten (sedezimal) auf dem Bildschirm auf.

M: Gefolgt von einer Adresse zeigt den Inhalt der Speicherzelle an, der durch Eingabe des neuen Wertes geändert werden kann. Mit CR wird das nächste Byte angezeigt.

D: Mit Start- und Endadresse eingegeben, überträgt es den Inhalt formatiert und mit Prüfsumme an den UART zur Speicherung auf Kassette oder Ausgabe auf TTY.

L: Lädt Daten vom UART, die mit „D" gespeichert wurden, und schreibt sie in die entsprechenden Speicherplätze. Bei „D" und „L" signalisiert eine LED, daß der Kassettenrecorder eingeschaltet werden muß.

B: Dieses Kommando setzt einen Unterbrechungspunkt in die entsprechende RAM-Adresse. Bei Erreichen wird der Programmablauf unterbrochen, alle CPU-Register werden abgespeichert und angezeigt.

E: Hiermit wird das Anwenderprogramm gestartet, und alle Register werden mit dem Inhalt des „Auslagerungs-RAMs" geladen.

S: Damit können Befehle schrittweise ausgeführt werden. Es werden jedesmal alle Registerinhalte angezeigt. Zum Starten des nächsten Befehls muß lediglich die CR-Taste betätigt werden. Es können alle Register- und Speicherinhalte modifiziert werden. Dieses Kommando wird hardwaremäßig realisiert, und zwar mittels eines nicht maskierbaren Interrupts.

Parvis Granmayeh

Rolf-Dieter Klein

S-100-System mit BASIC und schnellem Kassetten-Interface

S-100-Systeme haben den Vorteil, daß für sie eine große Auswahl an preiswerten Zusatzkarten zur Verfügung steht. Die Besonderheiten der beschriebenen Anlage sind ein 12-K-BASIC-Interpreter und ein schnelles Kassetten-Interface (bis 2400 Bd).

1 Hardware

Die Hardware besteht im Prinzip aus dem eigentlichen Rechner und der Peripherie. Der Rechnerteil gliedert sich in CPU-Karte (Cromemco-Z-80-CPU), Speicher und Interface-Karte. Die Peripherie setzt sich zusammen aus dem Datensichtgerät zur Ein- und Ausgabe der Programme und aus dem Kassettenrecorder.

1.1 CPU-Karte

Die CPU-Karte ist mit dem Mikroprozessor Z-80 aufgebaut und enthält die Anpassungs- und Treiberschaltung für den S-100-Bus. Der Mikroprozessor kann mit einem Takt von 4 MHz und – falls notwendig – mit WAIT-Zyklen betrieben werden. Letzteres ist für den Anschluß langsamer Speicher von Bedeutung. Auf der Karte ist außerdem eine Logik untergebracht, die dafür sorgt, daß bei einem Reset der Prozessor nicht an der ersten Speicherzelle beginnt, ein Programm abzuarbeiten, sondern an einer einstellbaren Adresse (*Power on Reset Jump*). Dieselbe Logik befindet sich nochmal auf der Interface-Karte, so daß sie hier eigentlich überflüssig ist.

1.2 RAM-Karten

Als Schreib/Lese-Speicher wurden zwei 16-KByte-Karten mit statischen RAMs gewählt. Für den Betrieb des BASIC-Interpreters sind bereits 12 K nötig. Für den Anwender stehen somit noch 20 K zur Verfügung.

1.3 Interface-Karte

Bild 1 zeigt die Blockschaltung der verwendeten Interface-Karte. Sie beinhaltet drei Serien-Ports mit einstellbarer Baudrate. Zwei davon werden für den An-

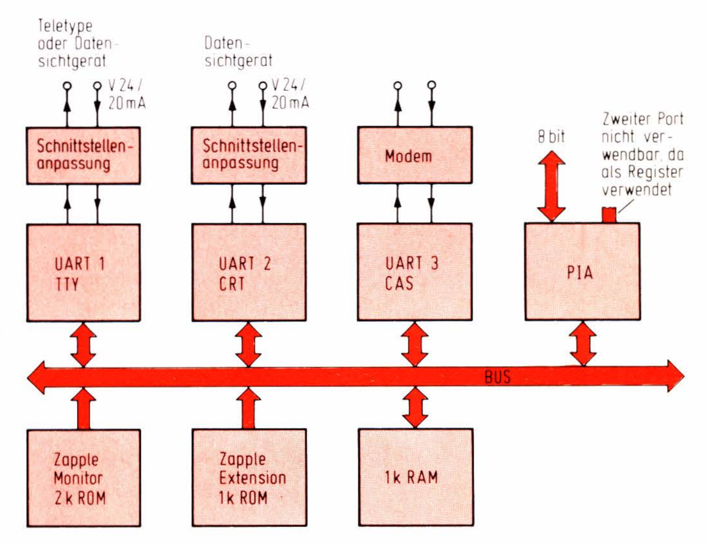


Bild 1. Blockschaltung der Interface-Karte SMB2 (Vertrieb: Datamega)

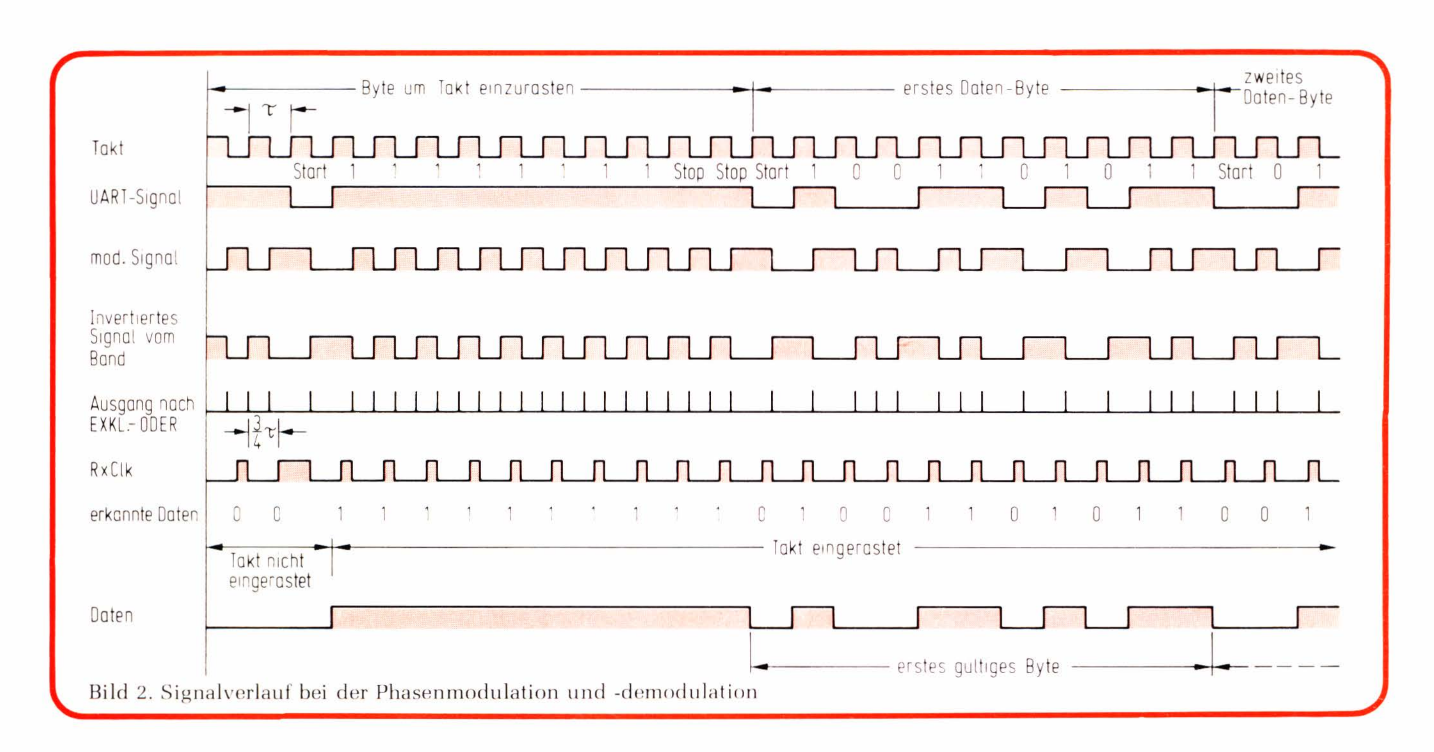


Bild 2. Signalverlauf bei der Phasenmodulation und -demodulation

Tabelle 1. Befehlssatz des Monitors

Befehl	Wirkung
A	Zuweisung von Lochstreifenleser/-stanzer oder Ausdruckeinheit von der Konsole
B	System sperren
C	Inhalt des Speichers mit dem Lesereingang vergleichen und Abweichung ausdrucken
D	Anzeigen eines Speicherbereichs (sedezimal)
E	Endezeichen ausgeben
F	Füllen eines Speicherbereichs mit einer Konstanten
G	Programm ausführen
H	Summe und Differenz zweier Sedezimalzahlen
I	Wird vom Benutzer definiert
J	Speichertest
K	Wird vom Benutzer definiert
L	Laden von Binärdaten
M	Speicherbereich verschieben
N	Nullen auf Stanzer geben
O	Wird vom Benutzer definiert
P	Zeicheneingabe in den Speicher von der Tastatur
Q	Ein- und Ausgabe über E/A-Port
R	Sedezimaldatei lesen (mit Prüfsumme, verschiebbar)
S	Speicherzelle mit neuen Daten überschreiben
T	ASCII-Äquivalent eines Speicherbereichs ausdrucken
U	Binär-Lochstreifen ausstanzen
V	Speicherbereiche vergleichen und Differenz ausdrucken
W	Sedezimaldatei (mit Prüfsumme) auf Stanzer geben
X	Anzeigen und Modifizieren aller Register (einschließlich Z-80-Spezialregister)
Y	String im Speicher suchen und Adresse anzeigen
Z	Höchste Speicheradresse anzeigen

schluß von Teletype bzw. Datensichtgerät benötigt. Der dritte Serienport ist an einen Modulator angeschlossen und für den Anschluß eines Kassettenrecorders vorbehalten. Es kann mit zwei verschiedenen Geschwindigkeiten auf Kassette aufgezeichnet werden, mit 1200 Bd und mit 2400 Bd. Dazu wird ein spezielles Aufzeichnungsverfahren verwendet. Es handelt sich dabei um die Phasenmodulation. Die Wirkungsweise zeigt *Bild 2*. Das Ausgangssignal des UART wird dabei mit Hilfe des eigenen Taktsignals (1:1) moduliert. Dies geschieht über eine Exclusiv-ODER-Verknüpfung (*Bild 3*). Das Ausgangssignal kann direkt auf die Kassette aufgezeichnet werden. Die Demodulation ist etwas komplizierter. Es muß ein synchroner Takt gewonnen werden. Dazu wird das Signal über zwei Monoflops geführt, von denen das eine auf die positive, das andere auf die negative Flanke reagiert. Es wird so ein Signal gewonnen, das bei jeder Flanke einen kurzen Impuls aufweist. Dieses Signal wird nun an ein weiteres Monoflop geführt, das auf eine Zeit von $^3/_4$ der Taktperiode eingestellt ist. Nach dem ersten H/L-Wechsel rastet dieser entstehende Takt ein. Der Takt wird dem UART zugeführt und steuert dort die Übernahme der Daten. Mit diesem relativ einfachen Modem kann auf einem normalen Kassettenrecorder eine Aufzeichnungsdichte von bis zu 2400 Bd erreicht werden.
Auf der Interface-Karte befindet sich außerdem noch ein Parallel-Port, von dem 8 Bits zur freien Verfügung stehen. Die Karte beinhaltet ein Monitor-Programm von 3 KByte, das einen recht komfortablen Betrieb zuläßt. Für allgemeine Verwendung befindet sich noch 1 KByte RAM auf der Karte.

2 Software

2.1 Zapple-Monitor

Tabelle 1 zeigt alle Befehle der 2-K-Grundversion. Mit Hilfe des Monitors kann man Programme eingeben, korrigieren, starten und abspeichern, wobei verschiedene Ausgabeformate möglich sind (binär oder Intel-Hex-Format). Ein zusätzliches 1-K-ROM gestattet es, EPROMs zu programmieren, wenn man über die sogenannte BYTESAVER-Karte (Cromenco) verfügt. Es handelt sich dabei um eine Karte, mit der man gleichzeitig acht EPROMs (zeitseriell) programmieren kann. Das ROM enthält ferner weitere Ein/Ausgabe-Routinen, unter anderem ein Programm, mit dem man im Blockformat auf Kassette aufzeichnen kann. Damit können größere Programme über den Recorder mit dem Assembler übersetzt werden. Vom Kassettenrecorder wird dabei immer ein ganzer Block in den Arbeitsspeicher geladen, der vom Assembler zeichenweise gelesen wird. Ist der Block leer, wird der nächste geladen usw.

2.2 Zapple-BASIC

Eine Liste aller verfügbaren Befehle zeigt *Tabelle 2*. Mit dieser BASIC-Version ist es möglich, mit einer

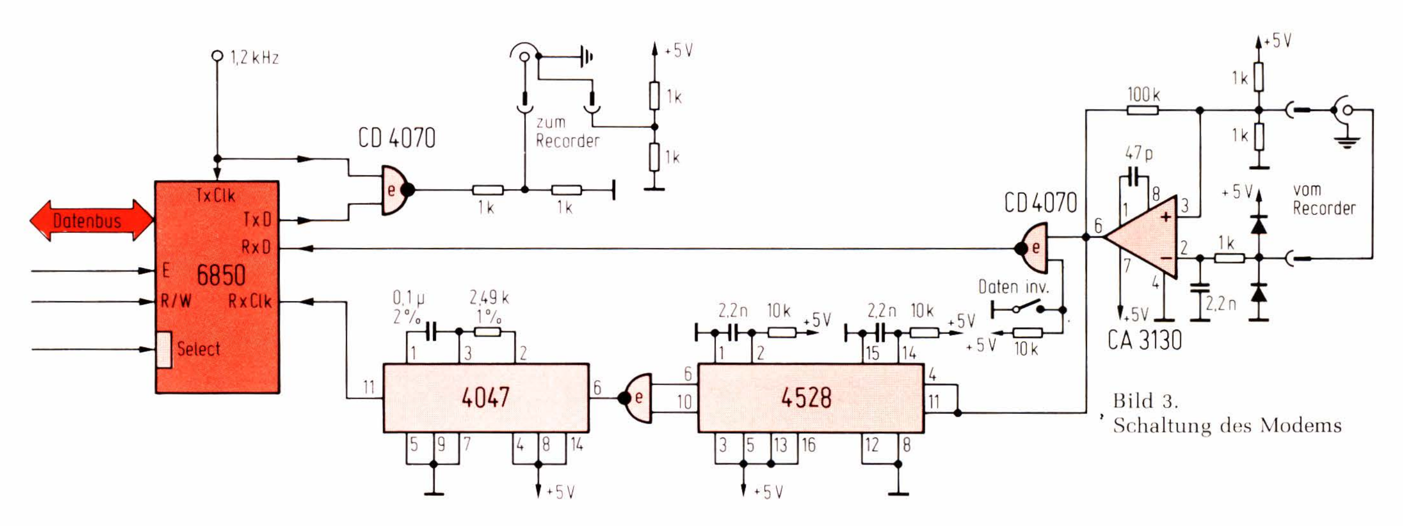


Bild 3. Schaltung des Modems

Genauigkeit von 12 Stellen zu rechnen. Der Bereich geht von $10^{-38}...10^{38}$. Es stehen zahlreiche Befehle für die Arithmetik zur Verfügung sowie ein besonderer Befehl zur Korrektur von Programmen (EDIT), mit dem es auf einfache Weise möglich ist, Tippfehler auszubessern. Der BASIC-Interpreter kann leicht an andere Systeme angepaßt werden, da am Anfang eine Sprungtabelle für E/A-Routinen steht. In diese Tabelle können die Sprünge zu den eigenen E/A-Routinen eingesetzt werden.

Literatur

[1] TDL SMB2 User Manual. Technical Design Labs.
[2] 12 K Zapple BASIC User Manual. Technical Design Labs.

Tabelle 2. BASIC-Befehlssatz

Befehl	Wirkung
ABS	Absolutwert
ALOAD, ALOADC, AMERGE, AMERGEC	Laden eines ASCII-Programms
AND	Logisch UND
ASAVE	Ausgabe des Programms in ASCII
ASC	Zeichen in numerischen Wert konvertieren
ATN	arctan-Funktion
AUTO	Automatische Zeilennumerierung
CALL	Aufruf für Maschinenprogramme
CHR$	Numerischen Wert in ASCII-Zeichen konvertieren
CLEAR	Alle Variablen löschen und Speicher für Strings reservieren
CONT	Programmausführung nach Unterbrechung fortsetzen
COPY	BASIC-Programmteil verschieben
COS	cos-Funktion
DATA	Definition von Konstanten
DEF	Benutzerfunktion
DELETE	Löschen
DIM	Dimensionierung von Matrizen
EDIT	Aufruf des Zeilen-Editors
ELSE	Sprunganweisung
END	Programmende
EXCHANGE	Austauschen von Variablen-Werten
EXP	Exponentialfunktion
FN	Benutzerfunktion
FNEND, FNFAC, FNRETURN	Mehrzeilige Funktionen
FOR	Beginn einer Schleife
FRE	Freier Speicher
COSUB	Unterprogrammaufruf
GOTO	Sprunganweisung
IF	Sprungtest
INP	Eingabe von E/A-Port
INPUT	Eingabe von der Tastatur
INSTR	String suchen
INT	Integer-Funktion
KILL	Speicherreservierung aufheben
LEFT$	Linker Teil eines Strings
LEN	Länge eines Strings
LET	Zuweisung
LINE INPUT	Erhöht die Flexibilität der Tastatureingabe
LIST	Programm ausdrucken (auf Konsole)
LLIST	Programm ausdrucken (auf Drucker)
LLVAR	Variable ausdrucken (auf Drucker)
LNULL	NUL-Zeichen auf Drucker ausgeben
LOAD	Programm vom Leser laden
LOADGO	BASIC-Programm lädt ein anderes Programm und übergibt die Steuerung
LOG	Natürlicher Logarithmus
LPOS	Position des Druckkopfes
LPRINT	Ausgabe auf Drucker
LVAR	Variable auf Konsole ausdrucken
LWIDTH	Druckerbreite einstellen
MID$	Mittelteil eines Strings
NEW	Programm und Variable löschen
NEXT	Rückkehr zum Schleifenbeginn
NOT	logisch NICHT
NULL	NUL-Zeichen an die Konsole ausgeben
ON	Mehrfach verzweigter Sprung
OR	logisch ODER
OUT	Ausgabe über E/A-Port
PEEK	Direkter Zugriff zum Speicher
POKE	Direktes Einschreiben in den Speicher
POS	Schreibposition der Konsole
PRECISION	Einstellen der Dezimalstellen
PRINT	Ausgabe über Konsole
PRINT USING	Formatanweisung
RANDOMIZE	Zufallsfunktion einstellen
READ	Daten von DATA-Zeile lesen
REM	Bemerkung
RENUMBER	Zeilennummern ändern (auch bereichsweise)
RESTORE	DATA-Zeiger rücksetzen
RETURN	Rückkehr vom Unterprogramm
RIGHT$	Rechter Teil eines Strings
RND	Zufallsfunktion
RUN	Programmstart
SAVE	Programmausgabe auf Peripheriegerät
SGN	Vorzeichen einer Variablen
SIN	sin-Funktion
SPC	Leerzeichen in der PRINT-Anweisung
SQR	Wurzel
STEP	Schrittweite in Schleifen
STOP	Programmausführung beendet
STR$	Numerischen Wert in ASCII-String konvertieren
SWITCH	Konsolenzuweisung ändern
TAB	Position des Druckkopfes in der PRINT-Anweisung einstellen
TAN	tan-Funktion
THEN	Sprunganweisung
TO	Sprunganweisung
TRACE	Zeilennummern der ausgeführten Befehle ausdrucken
USR	Benutzerfunktion
VAL	String in numerischen Wert konvertieren
WAIT	Status-Port abfragen
WIDTH	Schreibbreite der Konsole einstellen
HEXADECIMAL CONSTANT	Konstante
?	Entspricht PRINT
↑	Exponential-Operator
–	Minus
*	Mal
/	Geteilt durch
+	Plus
<	Kleiner als
>	Größer als
=	Ist gleich
CTRL U	Zeile löschen
CTRL C	Programm unterbrechen
CTRL X	Rückkehr zum Monitor
CTRL O	Konsolenausdruck unterbinden
CTRL R	Mehr Eingabezeichen
CTRL T	Ausdrucken der gerade ausgeführten Zeilennummer
CTRL S	Zeitweise Unterbrechung
CTRL Q	Wiederstart des Programms
RUBOUT	Vorhergehendes Zeichen löschen
,	Druckkopf zur nächsten TAB-Position
;	Druckkopf bleibt an derselben Stelle
:	Trennung für Mehrfachbefehle

Peter Schaltegger

TRS-80 – ein professioneller Hobbycomputer

Ende vergangenen Jahres stellte die amerikanische Firma Radio Shack, eine Abteilung der Tandy Corporation, einen „Jedermann-, Heim- oder Privat-Computer“ vor, der trotz seines geringen Preises (die Grundausführung mit Monitor und Kassettenrecorder ist schon für unter 1800 DM erhältlich) erstaunlich leistungsfähig ist. Mit den erhältlichen Zusatzeinheiten läßt sich der TRS-80 zu einem System ausbauen, das ohne weiteres kommerzielle Aufgaben übernehmen kann.

Das *Bild* zeigt die Grundausstattung. Das „Herz“ des Computers ist ein Z-80-Mikroprozessor. Er ist zusammen mit dem Benutzerspeicher (RAM), dem Programmspeicher (ROM) und der Ein/Ausgabe-Elektronik (I/O) auf einer Leiterplatte unterhalb der Tastatur untergebracht.

Die Bildschirmdiagonale des Monitors mißt 30,5 cm (12 Zoll) und erlaubt die Darstellung von 1024 Zeichen (16 Zeilen à 64 Zeichen) oder grafische Darstellungen mit einer Auflösung von 128 Punkten horizontal und 48 Punkten vertikal. Praktisch heißt das, daß der kleinste Punkt, der sich auf einem 12-Zoll-Bildschirm darstellen läßt, etwa 1,5 x 3 mm^2 mißt.

Durch Lösen der sechs Schrauben am Boden des Gehäuses können die beiden Gehäusehälften getrennt werden. Tastatur und Computerplatine sind Rückseite an Rückseite auf Nocken im Gehäuseboden gesteckt und können ohne Entfernen weiterer Schrauben aus dem Gehäuse gehoben werden. Die Computerplatine ist sehr sauber aufgebaut. Sämtliche Bauteile sind bezeichnet und können so schnell auf dem Schaltbild lokalisiert werden. Empfindliche Bauteile wie CPU, RAMs und ROMs sind auf Stecksockeln, alle anderen Bauteile sind direkt auf die Leiterplatte gelötet.

Dank der Verwendung von Low-Power-Schottky-TTL-Bausteinen erwärmt sich die Platine auch bei längerem Betrieb kaum. Lediglich das Netzteil, das außer dem Netztrafo ebenfalls auf der Computerplatine sitzt, strahlt etwas Wärme ab, ist aber so plaziert, daß die Kühlbleche direkt unter den Lüftungsschlitzen im Gehäuse liegen.

Schaltung

Der Schaltungsaufbau des TRS-80 entspricht dem eines üblichen Mikrocomputers mit CPU, Datenbus, RAM, ROM, Taktgeber und Ein/Ausgabeleitungen. Da aber Tastatur, Bildschirmspeicher und Video-Interface mit integriert sind, hat die CPU wesentlich mehr zu leisten als in vergleichbaren Systemen mit externen Terminals. Allein die Bedienung der Tastatur, deren Tasten über acht Adreß- und acht Datenleitungen bedient und softwaremäßig decodiert werden, beansprucht einige CPU-Zeit. Zudem läuft die CPU im TRS-80 „nur“ mit 2 MHz und nicht mit 4 MHz.

Dieses Konzept äußert sich natürlich in der Rechengeschwindigkeit. Das TRS-80-BASIC gehört aus diesem Grunde nicht zu den schnellsten Z-80-BASICs. Eine Erhöhung der Taktfrequenz ist aber nicht zu empfehlen, da dann verschiedene Probleme entstehen würden, wie z. B. Tastenprellen der Tastatur (zu schnelle Abfrage), und es müßten schnellere Speicher verwendet werden.

Als Schutz für die CPU und um auch am extern verfügbaren Bus genügend Leistung zur Verfügung zu stellen, sind sämtliche Adreß- und Datenleitungen über Bustreiber geführt, bevor sie andere periphere Bausteine erreichen. Bei Kurzschlüssen auf dem Bus-System nehmen also nur diese Treiber und nicht die CPU Schaden.

Ausbaufähigkeit

Wem die Grundausführung des TRS-80 nicht genügt, der hat nachträglich die Möglichkeit, sein Gerät zu erweitern und seinen Bedürfnissen anzupassen.

Da der eigentliche Computer zusammen mit der Tastatur in einem sehr kompakten Gehäuse untergebracht ist, wird für Speicher- (max. 48 KByte) und Peripherie-Erweiterungen ein zweites Gehäuse erforderlich. Diese Box, die im Design zum TRS-80 paßt und



Grundausstattung des Heimcomputers TRS-80 (erhältlich in den Tandy-Läden). Weitere Lieferfirmen: ABC-Computershop, 8 München 40, Schellingstr. 33; in der Schweiz: Comicro AG, 8003 Zürich, Badenerstr. 281

durch einfaches Stecken mit dem Computer verbunden werden kann, enthält z. B. weitere 16 K oder 32 K RAM, ein Zweifach-Kassetten-Interface, einen Floppy-Disk-Controller für vier Mini-Floppy-Disk-Laufwerke sowie ein Interface für den Anschluß eines Zeilen- oder „Bildschirm"-Druckers.

Das Erweiterungs-Interface enthält zusätzlich einen Echtzeit-Taktgeber, mit dem Zeiteinblendungen auf dem Bildschirm oder Drucker möglich werden, sowie einen weiteren freien Platz, in dem je nach Anwendung eine beliebige, zusätzliche Karte gesteckt werden kann.

Eine dieser Karten ist ein serielles Interface mit einer RS-232-Schnittstelle. Über dieses Interface können also auch Peripheriegeräte angeschlossen werden, die nicht im Radio-Shack-Programm enthalten sind.

Außerdem besteht die Möglichkeit, den TRS-80 über einen S-100-Adapter (Hersteller: Mini Mikro Mart Inc.) an den S-100-Bus anzuschließen.

Level-II-BASIC

Durch einen einfachen Umbau, der darin besteht, die beiden ROM-Chips (je 2 KByte) durch eine zusätzliche kleine Leiterplatte mit drei ROM-Chips (je 4 KByte) zu ersetzen, bietet der TRS-80 eine sehr leistungsfähige BASIC-Version, deren Befehlsvorrat und Editing-Funktionen aus der *Tabelle* zu ersehen sind. Neben den normalen BASIC-Befehlen sind darin auch Grafik-Befehle wie SET und RESET enthalten, mit denen direkt durch Angabe der X-Y-Koordinaten (SET(X,Y)) Punkte auf dem Bildschirm gesetzt oder gelöscht werden können. Ebenso sind bereits die Befehle zum Betrieb des Mini-Floppy-Disk-Laufwerkes enthalten.

Mini-Floppy-Disk-Laufwerk

Soll ein leistungsfähiges System aufgebaut werden, mit dem auch kommerzielle Ansprüche erfüllt werden können, so empfiehlt sich der Betrieb der „TRS-80-Minidisk". Diese Minidisketten haben eine Speicher-

Tabelle. Befehle und Eigenschaften von Level-II-BASIC

Variablen-Typen

$	String (0...255 Zeichen
%	Integer (–32768 to 32767)
!	Single precision (7,1 digit floating point)
#	Double precision (16,8 digit floating point)

Arithmetische Funktionen

ABS	PEEK
ATN	POINT
CDBL	POS
COBL	RANDOMIZE
CINT	RND
COS	SGN
CSNG	SIN
ERL	SPC
ERR	SQR
EXP	TAB
FIX	TAN
MEM	USR
INP	VARPTR
INT	
LOG	

Programm-Befehle

CMD	GOSUB	POKE
DEFDBL	GOTO	REM
DEFINT	IF	RESET
DEFSNG	LET	RESTOR
DEFSTR	NEXT	RESUME
DIM	ON ERROR GOTO	RETURN
END	ON...GOTO	SET
ERROR	ON...GOSUB	STOP
FOR	OUT	

Ein/Ausgabe-Befehle

INPUT	DATA	CLOAD	PRINT #
PRINT	SET	CSAVE	INPUT #
READ	RESET	INKEY$	PRINT USING

Kommandos

CLEAR	DELETE	LIST	NEW	TROFF
CONT	EDIT	LLIST	RUN	TRON

String-Funktionen

ASC	MID$
CHR$	RIGHT$
LEFT$	STR$
LEN	VAL

Disk-Befehle

E/A-Befehle		E/A-Funktionen	
CLOSE	LSET	CVD	LOF
DATA	OPEN	CVI	MKD$
FIELD	PRINT	CVS	MKI$
GET	PUT	EOF	MKS$
KILL	RESET	LOC	

Spezielle TRS-80-Befehle

SET	POINT	CLOAD	RANDOMIZE
RESET	INKEY	CSAVE	CMD

Editier-Befehle

A – Gehe zum Zeilenanfang
nC – n Zeichen ändern
nD – n Zeichen löschen
E – Ende des Editiervorgangs
H„Text" – Rest der Zeile löschen und „Text" einfügen
I„Text" – „Text" einfügen
L – Rest der Zeile drucken und zum Zeilenanfang gehen
Q – Alle Änderungen ignorieren
und einige andere

kapazität von 80 KByte und ermöglichen die Speicherung von Programmen und Daten nach verschiedener Organisation. Da ein Teil des DOS (*Disk Operating System*) bereits im Level-II-BASIC enthalten ist, ist dieses natürlich Voraussetzung zum Betrieb der Minidisks. Ebenso sind das Erweiterungs-Interface und ein minimaler Speicher von 16 K RAM erforderlich. Die Minidiskette ist in 35 Spuren à 18 Sektoren eingeteilt. Jeder dieser 630 Blöcke kann 128 Byte speichern. Für den Benutzer heißt das, daß er etwa 600 Adressen à 128 Zeichen auf einer Minidiskette speichern kann (einige Blöcke gehen verloren durch Inhaltsverzeichnisse oder sind nur für sequentielle Dateien zugelassen).

Peripherie

Wie es von einem richtigen Computer verlangt wird, können auch am TRS-80 verschiedene Drucker angeschlossen werden. Als einfachste und preisgünstigste Möglichkeit wird ein „Bildschirm"-Drucker angeboten, der ohne besonderes Interface direkt am Busstecker auch von Level-I-Computern angeschlossen werden kann. Durch Knopfdruck wird der Bildschirminhalt mit 2200 Zeichen/s auf ein ca. 10 cm breites Metallpapier „gebrannt". Dank der guten Auflösung des Druckers können auch grafische Darstellungen vom Bildschirm kopiert werden. Leider erfolgt die Aufzeichnung quer zur Papierlaufrichtung, so daß Listings oder Protokolle, die den Bildschirminhalt überlaufen, in mehreren Blöcken gedruckt werden müssen und so nicht sehr übersichtlich darzustellen sind.

Für höhere Ansprüche ist ein Centronix-Zeilendrucker lieferbar, der 60...110 Zeichen/s mit einer 5x7-Punkte-Matrix auf normales Papier drucken kann. Zum Betrieb dieses Zeilendruckers werden Level-II-BASIC und ein Erweiterungs-Interface vorausgesetzt.

Der TRS-80 läßt sich auch sehr gut für Textverarbeitung einsetzen: Es besteht z. B. die Möglichkeit, eine IBM-Selecterm-Schreibmaschine direkt am Parallel-Port des Erweiterungs-Interface anzuschließen und direkt durch die Befehle LPRINT und LLIST zu steuern. Mit entsprechendem Kugelkopf eingesetzt, lassen sich auch deutsche Texte mit Umlauten schreiben.

Der TRS-80 bietet mit seinen Zusatzeinheiten eine Vielfalt von Anwendungsmöglichkeiten: vom einfachen Spielcomputer bis zum kommerziell einsetzbaren System. Aber trotz dieser Ausbaumöglichkeiten muß man die Grenzen eines solchen Systems sehen. Vor allem im kommerziellen Einsatz müssen Rechengeschwindigkeit und Speicherkapazität genau betrachtet werden, um nicht Gefahr zu laufen, seine Zeit mit Warten vor dem Bildschirm zu verschleudern. Ist man sich aber dieser Grenzen bewußt, so wird man viel Freude damit haben, denn obschon das System der untersten Preisklasse zuzuordnen ist, läuft es zuverlässiger als viele seiner größeren Brüder.

PET, der Wunderknabe

Die Mikrocomputer-Familie

Der PET 2001 (*Bild 1*) ist strenggenommen eigentlich „nur" der Nachfolger eines ebenfalls schon recht verbreiteten Mikrocomputers, des KIM-1. Beide Geräte enthalten den gleichen Mikroprozessor, nämlich den MCS 6502 von MOS Technology (kürzlich von Commodore übernommen).

Der Entwickler des MCS 6502 heißt Chuck Peddle; aus seiner Hand stammen auch der KIM und der PET 2001. KIM heißt Keyboard Input Monitor; er erlaubt die Eingabe von Befehlen in sedezimaler Maschinensprache, die Anzeige von Adressen und Daten auf einem sechsstelligen Siebensegment-Display und den Anschluß eines ASCII-Fernschreibers oder Terminals über eine 20-mA-Schnittstelle. Für die Verbindung mit der „übrigen Welt" sind darüber hinaus 15 I/O-Ports vorhanden. Man schätzt, daß es allein in Europa heute knapp 6000 KIM gibt – eine bemerkenswerte Zahl, wenn man bedenkt, daß zur Programmierung in Maschinensprache einige Erfahrung notwendig ist.

Der „Sohn" des KIM-1, der PET 2001, ist hier besser dran; er gestattet die Eingabe von direkten Befehlen und Programmen in der „Sprache" BASIC (Beginners' All-Purpose Symbolic Instruction Code). Das bedeutet, er versteht einfache Befehlsworte in englischer Sprache; z. B. PRINT "I AM A PET", was zum Ausdruck der Worte I AM A PET führt.

Es sollte vielleicht noch bemerkt werden, daß es auch für den KIM-1 ein BASIC-Programm gibt, das allerdings natürlich nur mit einiger Speichererweiterung sinnvoll ist; der Original KIM besitzt 1,1 KByte RAM. Das „Tiny BASIC" für den KIM wurde von Tom Pittmann entwickelt, der zu der „San Francisco Community" gehört und diese Software für 5 Dollar in Lochstreifenform vertreibt.

Das BASIC-Programm des PET befindet sich dagegen in einigen ROMs und belegt 8 KByte.

Die Tatsache, daß der PET 2001 zu den schnellsten derzeit erhältlichen Tischcomputern gehört, und das trotz seines Preises unter 3000 DM, ist vor allem auf die Eigenschaften des verwendeten Mikroprozessors zurückzuführen. Ohne auf Details einzugehen, seien doch einige Punkte für denjenigen erwähnt, der sich schon etwas näher mit der Materie beschäftigt hat.

Die hohe Geschwindigkeit kommt vor allem durch das „Pipelining"-Speicherzugriffs-Verfahren zustande, durch das die meisten Zwei-Byte-Befehle auch tatsächlich nur zwei 1-µs-Maschinenzyklen benötigen (bei einem 1-MHz-Steuerquarz). Doch dies nur nebenbei; beim Programmieren in BASIC braucht man sich darum ja kaum zu kümmern.

Das PET-Grundkonzept

Heimcomputer wie der PET sind strenggenommen eine Weiterentwicklung programmierbarer Taschen- oder Tischrechner. Die Hersteller haben in den letzten Jahren erkannt, daß bei programmierbaren Taschenrechnern, wie etwa dem HP-67 oder dem TI-59, kaum noch eine weitere Perfektionierung sinnvoll sein kann. Die Grenzen einer weiteren Funktions-Expansion sind wegen der rein numerischen Tastatur heute nahezu schon erreicht, und selbst wenn, wie beim TI-59 mit einem zusätzlichen Drucker eine alphanumerische Ausgabe z. B. von Texten oder Formelzeichen möglich ist, so ist doch die alphanumerische Eingabe extrem umständlich: Sie muß nämlich in Form einer zweistelligen Code-Zahl für jeden Buchstaben erfolgen. Ferner erlaubt ein Taschenrechner nicht ohne weiteres den Anschluß externer Geräte, wie etwa zusätzlicher Speicher, Digital-Meßgeräte oder ganz allgemein die Steuerung externer Geräte. Zwar besitzt etwa der HP-97 S mittlerweile eine BCD-Schnittstelle, aber von einem komfortableren Bus-System wie dem IEC- oder dem IEEE-Bus (der PET besitzt letzteren) ist man noch weit entfernt.

Eine echte Leistungserhöhung solcher Rechner ist also nur möglich, wenn man alphanumerische Ein- und Ausgabemöglichkeiten vorsieht. Beim PET geschieht dies mit einer Tastatur (*Bild 2*), die etwas kleiner als die einer Schreibmaschine ist und die – rechts daneben – durch eine kleine numerische Tastatur ergänzt wurde. Die Eingabe von großen *und* kleinen Buchstaben ist nicht ohne weiteres durch die schlichte Betätigung einer „Shift"-Taste möglich, da normalerweise die zweite Tastenfunktion entweder Steuerbefehle oder grafische Symbole enthält. Normalerweise schreibt man also ausschließlich mit Großbuchstaben; das ist aber auch bei fast allen anderen Computersystemen und Terminals der Fall, soweit sie nicht der Textverarbeitung dienen.

Was den PET von anderen (erstaunlicherweise sogar teureren) Heimcomputern unterscheidet, ist die Tatsache, daß sowohl ein (wenn auch nur Schwarzweiß-) Monitor sowie ein Kassettenrecorder zur Speicherung von Programmen und Daten bereits eingebaut ist. Dies ist durchaus wesentlich, denn die Benut-

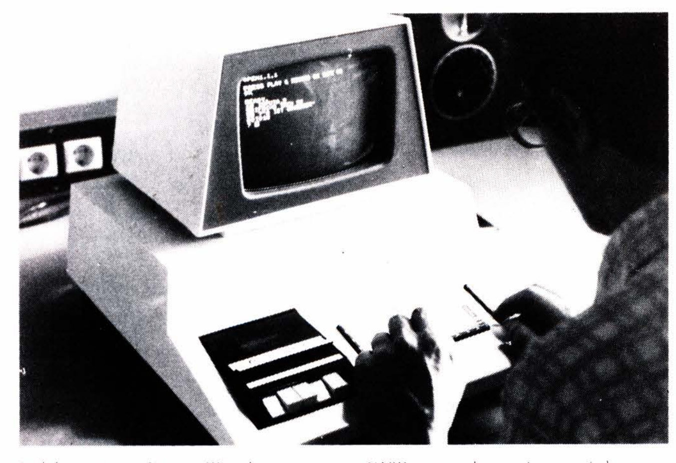


Bild 1. Der Basic-Tischcomputer PET 2001 hat einen Schwarzweiß-Monitor und einen Kassettenrecorder als Programmspeicher bereits eingebaut

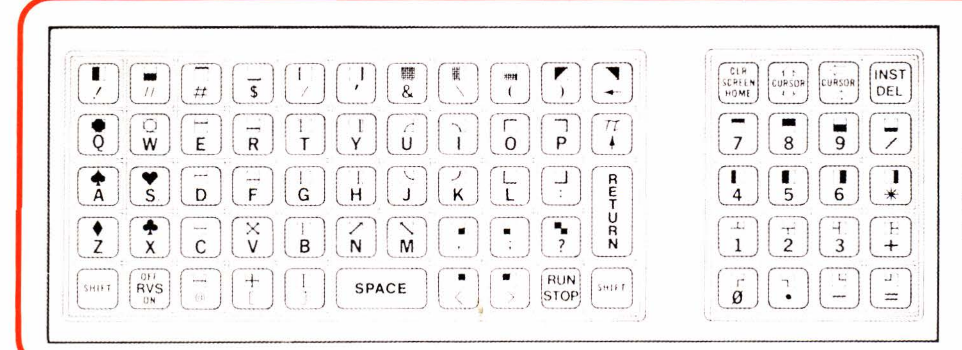


Bild 2. Das PET-Tastenfeld ist etwas kleiner als das einer Schreibmaschine, bietet aber einiges an grafischen Symbolen

zung des Computers wird dadurch von Absprachen mit anderen Familienmitgliedern unabhängig, die das Fernsehgerät manchmal auch als solches und nicht als Computer-Monitor benützen wollen.

Das ist drin...

Wie gesagt, der PET 2001 enthält also zunächst einmal eine ASCII-Tastatur, einen Schwarzweiß-Monitor und einen Kassettenrecorder, der mit einer Geschwindigkeit von etwa 600 Baud arbeitet; es handelt sich dabei übrigens um ein nur geringfügig umgebautes, handelsübliches Billig-Modell, bei dem ein Teil des Batteriekastens abgesägt wurde, um Platz zu sparen.

Das Herz des PET ist aber eine relativ große Computer-Platine, die die CPU, also den Mikroprozessor 6502, die ROM- und RAM-Speicher-IS sowie die notwendigen Input/Output-Bausteine enthält (*Bild 3*). Diese Platine ist *nicht* mit dem KIM-1 identisch! Der ROM-Speicher beinhaltet insgesamt 14 KByte, bestehend aus dem BASIC-Interpretierer (8 K), dem Betriebssystem (4 K), einem recht ausgeklügelten Prüfprogramm (1 K) und dem Maschinensprache-Steuerprogramm (1 K). Der Monitor erlaubt die Anzeige von bis zu 1000 Zeichen in 25 Zeilen zu je 40 Zeichen. Jedes Zeichen kann wahlweise aus einem üblichen ASCII-Zeichen (Buchstaben, Ziffern, Satzzeichen) oder einem Grafik-Symbol bestehen.

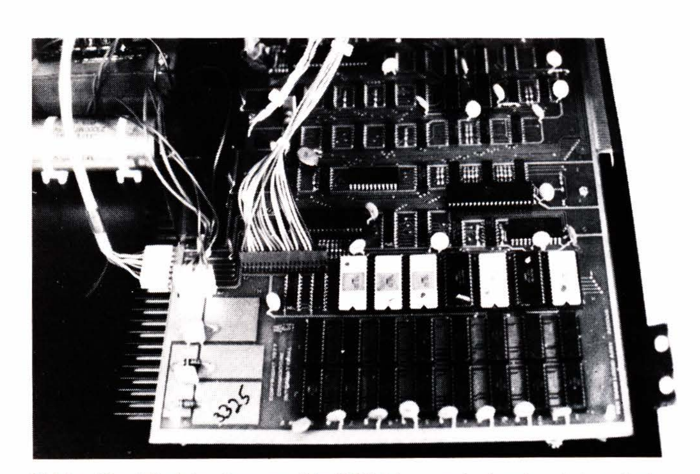


Bild 3. Ein Blick ins Innere. Die PET-Mutterplatine ist nicht mit dem KIM-1 identisch, sondern benützt nur den gleichen Mikroprozessor (Fotos: H. Feichtinger)

...und das kann der PET

Ja, zunächst einmal „kann" der PET BASIC-Befehle verstehen. Nun gibt es natürlich recht unterschiedliche BASIC-Versionen; zum Glück ist das Commodore-BASIC ein recht komfortables. Neben den üblichen Befehlen versteht er auch z. B. bedingte Programmsprünge (ON...GOTO und ON...GOSUB), die wichtigsten mathematischen Funktionen und logische Entscheidungen (AND, OR, NOT) und besitzt auch eine eingebaute Software-Uhr (TI $). Bemerkenswert ist auch die große Zahl von Input/Output-Anweisungen für verschiedene, einzeln adressierbare periphere Geräte, wie etwa Drucker (ein solcher wurde auf der Hannover-Messe vorgestellt und soll etwa 2000 DM kosten), Kassettenrecorder oder beliebige andere Geräte, die einen IEEE-Bus-Anschluß besitzen. Die Befehle OPEN und CLOSE dienen zum Eröffnen und Schließen eines logischen File (0...255), und auch für Zeichenketten (Strings) sind einige recht komfortable Anweisungen vorhanden, z. B. zur Bestimmung der String-Länge, zur ASCII-Code-Umwandlung oder zur Zusammenfügung mehrerer Strings zu einem.

Mit den Befehlen PEEK und POKE ist es möglich, Daten und Befehle in Maschinensprache, allerdings umgeformt in dezimale Form, auf den Bildschirm zu bringen oder zu speichern. USR gestattet zusammen mit SYS schließlich den Aufruf von Programmen in Maschinensprache.

Können Sie BASIC?

Wenn nein, müssen Sie es leider lernen. Dies ist allerdings mit Hilfe des PET relativ einfach; angeblich wird nämlich bald stets eine BASIC-Lehrkassette mitgeliefert. Das mitgelieferte „Handbuch" (beachten Sie bitte die Anführungszeichen) ist jedenfalls in keiner Weise geeignet, nach Erhalt des PET mit diesem Wunderwerk moderner Technik zurechtzukommen, beinhaltet es doch nicht einmal eine zusammenhängende Auflistung aller Steuer- und Programmbefehle nebst einer Übersicht, wann Klammern oder Anführungsstriche zu setzen sind. Die mitgelieferten schriftlichen Unterlagen beinhalten nämlich nicht die geringsten Programm-Anwendungsbeispiele!

Wer also in dem schönen, bunten Prospekt von den vielfältigen Möglichkeiten des PET begeistert war und dachte, gleich bei Anlieferung nun Kartenspiele oder

sonstige Programme durchführen zu können, wird bitter enttäuscht sein. Obwohl es sicher nur zwei Mark gekostet hätte, verzichtete Commodore leider darauf, wenigstens einige Demonstrations-Programme auf einer Kassette mitzuliefern. Das einzige, was man in den ersten Minuten nach dem Auspacken aus dem Wunderding herausholen kann, ist schlicht und einfach die Fehlermeldung „SYNTAX ERROR"...

Übrigens enthält das Handbuch auch einige Angaben über „BASIC Bugs" – Würmer in der ROM-Software also; z. B. wird die Folge IF F OR I = 10 fälschlich als IF FOR I = 10 interpretiert und führt zu einer Fehlermeldung. Einige solcher Fehler werden in einer verbesserten Software ausgemerzt; neue ROMs sind von Commodore in Kürze erhältlich. Dazu gehören auch Fehler wie z. B., daß SPC (0) nicht Null, sondern 256 Leerräume liefert, daß Direkt-Befehle ignoriert werden, die mit einem Doppelpunkt beginnen, und daß Felder mit mehr als 255 Elementen nicht möglich sind.

H. Feichtinger

Knipsen statt drucken – ein Tip, der Geld spart

Das klassische Dokumentationsmittel in der elektronischen Datenverarbeitung ist – auch für den Hobbycomputer-Freund – der Drucker. Aber Drucker sind noch immer teuer, und wer den Bildschirm-Inhalt seines Rechners oder Sichtgerätes nur gelegentlich festgehalten zu sehen wünscht, der fährt mit der Bildschirmfotografie billiger. Nach Versuchen mit verschiedenen Aufnahmeformaten und Kameratypen hat sich der Autor für folgendes Verfahren entschieden: Kleinbildkamera mit Teleobjektiv 135 mm Brennweite, zwei Zwischenringe, Film 15 DIN, Belichtungszeit bei Blende 8 eine Sekunde.

Wenn man darauf achtet, daß Bildschirm-Hauptebene und Filmebene möglichst parallel zueinander verlaufen, dann stören wegen der langen Brennweite die durch Bildschirmkrümmung entstehenden Verzerrungen kaum mehr. Sind sehr viele helle Zeichen auf dem Schirm, empfiehlt es sich, die Belichtungszeit auf die Hälfte zurückzunehmen. Die Fotos des Programms „Wobbel-PET" sind beispielsweise auf diese Art entstanden (Aufnahmedaten: Exa II mit Exaprox-Zwischenringen b und c, Schacht Travenar 1 : 3,5/135 R auf Agfapan 25 Professional; Entwicklung im Standard-Tank des „Fotohändlers um die Ecke").

Ein sehr billiges Verfahren – aber es hat einen Nachteil: Man muß warten, bis die Bilder entwickelt sind. Wer es eiliger hat und bereit ist, dafür etwa die doppelten Kosten und eine etwas geringere Bildqualität in Kauf zu nehmen (die für den Hausgebrauch aber allemal ausreicht, wenn man nicht gerade schöngedruckte wissenschaftliche Werke damit illustrieren lassen will), für den kann die Sofortbild-Fotografie empfohlen werden. Nach Versuchen mit anderen Verfahren benutzt der Autor hierfür eine Kamera Polaroid SX-70 II mit dem SX-70-Farbfilmpack. Diese Kamera ist übrigens mit einer elektrischen Fernauslösung versehen und kann über einen Schalttransistor oder ein empfindliches Relais auf dem Umweg über den Benutzer-Port vom Hobbycomputer selbst gestartet werden.

Erforderlich in jedem Fall: ein solides Stativ. Die *Bilder 1* und *2* zeigen die Auflistung und ein Beispiel eines Programmes, das die Zahl der Bilder berechnet, ab der ein Drucker billiger ist als Fotografieren.

Hans-Georg Joepgen



Bild 1. Wer nur gelegentlich den flüchtigen Inhalt auf dem Schirm seines Hobbycomputers in dauerhafterer Form dokumentiert zu sehen wünscht, für den kann Fotografieren billiger sein als Drucken: Das per Sofortbild-Fotografie dargestellte Programm „Kostenlupe" versetzt den PET und andere BASIC-Maschinen in die Lage, den Steckenpferdreiter vor kostenträchtigen Fehlentscheidungen zu schützen. Wenn der Rechner „Kostenlupe" geladen hat und gestartet wird, erfragt er alle Angaben, die er benötigt, und teilt dann mit, wo im speziellen Fall die „Schallgrenze" in den Anschaffungs- und Betriebskosten von Fotozubehör und Drukker liegt

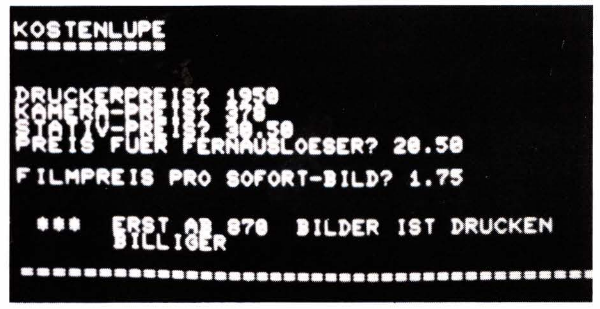


Bild 2. Beispiel einer Berechnung, ab wann ein Drucker rentabel ist

Der Mikrocomputer KIM-1 verfügt, wie an anderer Stelle in diesem Heft besprochen, über kaum mehr als 1 KByte RAM-Speicherkapazität. Um so erstaunlicher ist es, daß der KIM einen für Übungszwecke schon recht brauchbaren Schachpartner abgibt; ein entsprechendes Programm wurde von Peter Jennings, Toronto/Kanada, entwickelt.

Herwig Feichtinger

KIM spielt Schach

„Microchess", Mikro-Schach, wurde für den Betrieb einer Mikrocomputer-Minimalkonfiguration entwickelt. Das Programm wurde so ausgelegt, daß es auch die im KIM-Monitor enthaltenen Unterprogramme ausnützt; dadurch ist eine Verwendung in anderen, auf dem Mikroprozessor 6502 basierenden Systemen nicht ohne weiteres möglich, es sei denn, man implementiert die benötigten KIM-1-Routinen und ändert gegebenenfalls die Unterprogramm-Aufruf-Adressen im Schachprogramm.

All das ist aber bei Verwendung des KIM-1 nicht erforderlich. Am einfachsten besorgt man sich die relativ preiswert erhältliche Kassette (z. B. AB-Computershop, Schellingstraße 33, 8000 München 40), lädt sie mit dem bereits im KIM-1 enthaltenen Interface, und das Spiel kann beginnen.

1 Laden des Programms

Aufgrund der Speicherbereichs-Verteilung im KIM-1 ist es notwendig, das Programm in zwei Abschnitten zu laden. Beide Abschnitte besitzen die Kennziffer C1, die in die Speicherzelle mit der Adresse 17F9 zu schreiben ist. Vergessen Sie aber nicht – das KIM-Handbuch weist leider nicht deutlich genug darauf hin – die Daten 00 in die Adresse 00F1 zu laden, um den richtigen Prozessor-Status sicherzustellen. Der NMI-Interrupt-Vektor (00, 1C) wird an den Adressen 17FA und 17FB abgelegt. Hat man dies alles getan, so braucht man nur noch auf die Adresse 1873 zu gehen; das ist die Startadresse des im KIM-Monitor enthaltenen Programms zum Lesen des Magnetbandes.

Jetzt drückt man die Taste „GO" auf dem KIM-Tastenfeld und startet den Kassettenrecorder. Damit werden zunächst alle Daten und Programmteile im Adressenbereich 0000...03FF geladen, der den acht 2102-Chips auf der KIM-Platine zugeordnet ist. Sobald die Siebensegment-Anzeige wieder aufleuchtet (0000 D8), geht man wieder an die Adresse 1873, drückt „GO" und startet den Kassettenrecorder, natürlich ohne ihn vorher zurückzuspulen, abermals. Jetzt werden die Daten im Adressenbereich 1780...17E5 vom Band geladen.

Trat beim Lesen des Bandes kein Fehler auf, so zeigt das Display nach kurzer Zeit (der zweite Abschnitt ist wesentlich kürzer als der erste) wieder 0000 D8 an. Mit der Taste „GO" wird an dieser Adresse 0000 nun das Programm gestartet; es ist aber ratsam, vorher noch sicherheitshalber „RS" zu drücken. Zunächst ändert sich an der Anzeige nichts; die Taste C sorgt aber für die Anzeige CCCC CC auf dem KIM-Display.

2 Bedeutung der Tasten auf dem KIM

Ist das Programm einmal mit der GO-Taste gestartet, so haben die Tasten des KIM-1 zum Teil eine andere Bedeutung, als man das gewöhnt ist. Wie gerade beschrieben, zeigt das Display beim Drücken der Taste C in allen sechs Stellen ein C. Das bedeutet, daß das „interne Schachbrett", mit dem sich der KIM die Stellung der Figuren merkt, auf die Anfangsstellung rückgesetzt wurde.

„C" bringt alle Figuren in ihre Anfangsstellung.

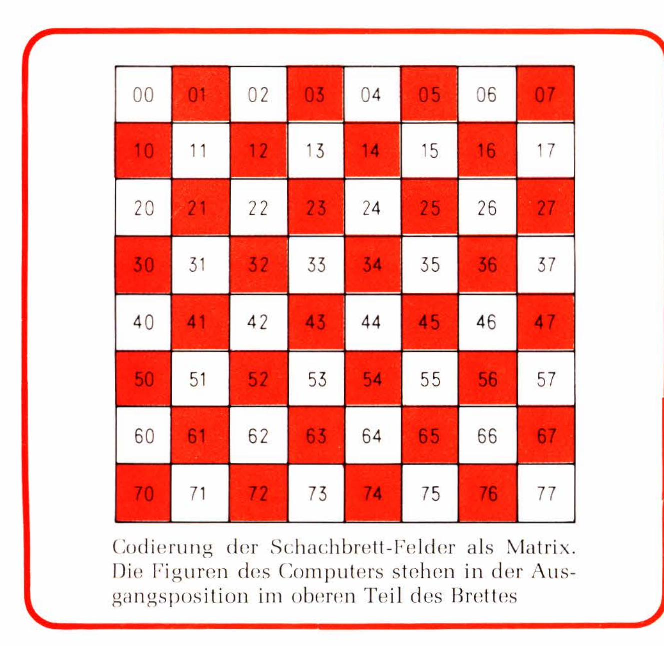
Wie bringt man dem KIM nun eigentlich bei, von wo nach wo eine Figur bewegt wird? Und wie zeigt der KIM seinen Zug an?

Das Schachbrett wird zu diesem Zweck erst einmal als Matrix betrachtet. Drücken Sie gleich einmal die Taste „PC". Auf dem Display erscheint 0F 13 33.

„PC" (Play Chess) läßt den KIM einen Zug machen.

Die Anzeige 0F 13 33 bedeutet nun: Der Computer zieht die Figur, die auf dem Feld mit der Nummer 13 steht, auf das Feld 33, und zwar – da der KIM das Spiel begann – mit einer weißen Figur.

Das *Bild* zeigt die Numerierung der Schachbrett-Felder. Im Gegensatz zu der sonst üblichen Feldbezeichnung 1...8 und A...H wird hier die Matrix nur mit den Zahlen 0...7 gekennzeichnet. Es wäre zwar möglich, eine entsprechende Umrechnung noch im Programm vorzunehmen; das ist aber wegen der begrenzten Speicherkapazität im KIM-1 nur mit zusätzlichen Speicherkarten durchführbar.



Codierung der Schachbrett-Felder als Matrix. Die Figuren des Computers stehen in der Ausgangsposition im oberen Teil des Brettes

Die Figuren des Computers sind stets im oberen Teil des Schachbretts (00...17) beheimatet.

Sie können jetzt Ihren Gegenzug eingeben – aber halt, was bedeuten eigentlich die beiden vorderen Stellen der Anzeige?

Die erste Stelle (von links) gibt an, wessen Figur bewegt werden soll. Eine Eins bedeutet: Die Figur, die auf dem „Von"-Feld, also mit den mittleren beiden Stellen des Display, angezeigt wird, ist Ihre Figur. Eine Null bedeutet dagegen, daß das „Von"-Feld eine Figur des Computers enthält. Sollte ein F in dieser Stelle aufleuchten, so bedeutet das: das „Von"-Feld beinhaltet keine Figur, z. B. wenn ein Zug bereits vollzogen wurde.

Probieren wir das gleich einmal aus. Drücken Sie die Tasten 6 3 4 3; Sie ziehen also Ihren Bauern.

Die Anzeige beinhaltet jetzt 1F 63 43; die Eins bestätigt, daß Sie wirklich eine eigene Figur ziehen. Drückt man jetzt die Taste F, so wird dieser Zug auch auf dem internen Schachbrett des KIM ausgeführt, also verifiziert. Die erste Stelle der Anzeige wechselt jetzt von 1 auf F; denn nach der Ausführung des Zuges ist das „Von"-Feld natürlich leer.

Tabelle. Figuren-Code und Speicheradresse

Code	Figur	Adresse der Comp.-Fig.	Ihre Figur
0	König	0050	0060
1	Dame	0051	0061
2	Turm	0052	0062
3	Turm	0053	0063
4	Läufer	0054	0064
5	Läufer	0055	0065
6	Springer	0056	0066
7	Springer	0057	0067
8...F	Bauern	0057...005F	0067...006F

„F" führt den gerade eingegebenen Zug auf dem internen Schachbrett aus.

Noch eine Anmerkung dazu: Die Taste F brauchen Sie nur bei eigenen Zügen zu betätigen; der Computer zeigt nach der Ausführung seiner Züge in der ersten Stelle kein F, obwohl auf dem internen Schachbrett die Figur bereits bewegt wurde.

Wenn jetzt wieder die Taste „PC" gedrückt wird, macht der Computer einen Zug, und das Spiel wird bis zum bitteren Ende fortgesetzt.

Sicher wollen Sie aber manchmal wissen, was für eine Figur da gerade gezogen wird; das wird verschlüsselt von der zweiten Stelle der Anzeige dargestellt. Die *Tabelle* nennt den Geheimcode.

Fehler bei der eigenen Eingabe lassen sich solange korrigieren, wie die Taste „F" noch nicht gedrückt wurde; man braucht nur – nach einer falschen Eingabe – die vier richtigen Ziffern noch einmal einzutasten und dann „F" zu drücken.

Noch eine Taste fehlt uns in unserer Aufstellung:

„E" vertauscht Weiß und Schwarz.

Nach dem Drücken der Taste „E" erscheint im Display EEEE EE, und die Figuren des Computers werden mit Ihren vertauscht. Aber, Vorsicht: Der Computer spielt nach wie vor so, als wären seine Figuren auf dem oberen Teil des Schachbrettes beheimatet! Sie müssen also sozusagen die Numerierung der Matrix-Felder umdrehen.

Wenn abwechselnd „PC" und „E" gedrückt werden, spielt der KIM eine – zuweilen recht interessante – Partie gegen sich selbst, aber natürlich immer die gleiche.

3 Besondere Züge

Leider ist der KIM-1 vor allem wegen seines begrenzten Speicherumfanges kein Universalgenie. Er beherrscht daher zwar die Grundregeln, nicht aber spezielle Züge des „Spiels der Könige".

Rochade

Die Rochade muß in zwei Teilzüge zerlegt werden. Nach jedem Teilzug ist die Taste F zu drücken; „PC" aber erst nach dem zweiten Teilzug.

Der Computer weiß nicht, wie's geht und führt daher selbst keine Rochade durch. Wenn Sie Anhänger der Gerechtigkeits-Ideologie sind, verzichten Sie daher besser auch selbst auf diesen Spezialzug.

En Passant

Um mit dem Bauern „im Vorübergehen" zu schlagen, müssen Sie auch diesen Zug in zwei Teilzüge zerlegen. Zunächst rücken Sie Ihren Bauern nur so weit, daß die feindliche Figur geschlagen wird; drücken Sie hierbei nur „F", nicht aber „PC". Dann schiebt man den Bauern weiter, drückt „F" und schließlich „PC". Auch dieser Spezialzug wird vom Computer nicht beherrscht, und gerechterweise wird man daher auf ihn meist verzichten.

Bauer wird zur Dame

Wenn ein Bauer bis zur vordersten gegnerischen Reihe gezogen wurde, wird er zur Dame. Das geht weit über KIM's Horizont, und man muß daher das Schachprogramm mit der ST-Taste (Stop) verlassen. Dies ist nur bei leuchtender Anzeige zulässig (!). Die Tasten haben jetzt wieder die im KIM-Handbuch beschriebene Bedeutung. Jetzt wird's kompliziert: Zunächst muß der Bauer auf seiner alten Position „ausradiert" werden. Schauen Sie also erst einmal in der Tabelle nach, wer wo steht. Gehen Sie an die Adresse 0068 und suchen Sie, welcher Bauer auf der jeweiligen Matrix-Position steht. Laden Sie die Daten CC in diese Speicherzelle, um dem Computer mitzuteilen, daß diese Figur beseitigt wurde.

Jetzt müssen wir nur noch dafür sorgen, daß eine Dame am gewünschten Schachbrett-Feld „geboren" wird. Gehen wir also an die Adresse 0061 und geben dort als neue Daten das jeweilige Matrixfeld ein. Mit den Tasten PC (jetzt „Program Counter") und GO kann man nun in das Schachprogramm zurückkehren.

Ähnlich muß verfahren werden, wenn es dem Computer gelingt, einen Bauern bis zur achten Reihe (7 als Microchess-Matrixzeile) vorzuschieben; die entsprechenden Adressen für die Figuren des KIM sind ebenfalls der Tabelle zu entnehmen. Zugegeben, etwas umständlich, aber wie schon gesagt – was ist schon 1 KByte Speicherplatz?

Es gibt in den oben erwähnten Spezialfällen immer zwei Möglichkeiten: Entweder gibt man dem Computer durch manuelles Eingreifen in das Programm (Rückkehr in den Monitor und Ändern der Daten) künstlich die fehlende Intelligenz, oder aber man hält sich selbst an etwas vereinfachte Schachregeln und verzichtet z. B. darauf, den Bauern in der achten Reihe in eine Dame umzuwandeln.

4 Drei Intelligenz-Stufen

Durch gezieltes Ändern des Programms läßt sich die Spielintelligenz des KIM ändern. Ist das Programm von der Kassette geladen, so spielt der Computer im Normal-Modus; er braucht für einen Zug (außer während der Eröffnung, die spielt er aus dem Gedächtnis) durchschnittlich etwa hundert Sekunden. Im Blitz-Modus müssen an die Adressen 02F2 bzw. 018B die Daten 00 bzw. FB geladen werden; ein Zug benötigt dann nur noch etwa zehn Sekunden. „Super-Blitz" drückt diese Zeit sogar auf etwa drei Sekunden herab; laden Sie dann an die gerade genannten Adressen die Daten 00 und FF. Will man wieder in den Normal-Modus zurückkehren, so sind diese Daten durch 08 bzw. FB zu ersetzen.

Wahrscheinlich wird es Ihnen wie dem Verfasser dieser Zeilen gehen: Man schafft es zunächst nicht, ein Spiel ganz „durchzuziehen", weil man sich nicht so schnell an die etwas ungewohnte Matrix-Felder-Bezeichnung gewöhnt und vielleicht auch irgendwann Eingabefehler macht, die man erst später bemerkt.

Gelingt es schließlich aber, den Computer matt zu setzen, so quittiert er dies mit der Anzeige FF FF FF.

5 Keine Chance gegen den Computer?

Nein, ganz so schlimm ist es nicht. Der KIM ist zwar, wie gesagt, kein Universalgenie, aber auch kein blutiger Anfänger. Eine Standard-Eröffnung spielt er zunächst aus dem Gedächtnis (das merkt man an der extrem schnellen „Berechnung" der ersten Züge), und alle weiteren KIM-Züge werden nach einer festgelegten Routine unter allen Möglichkeiten beim aktuellen Spielstand ausgesucht.

Die Bewertungskriterien für den besten Zug sind:

- Beweglichkeit der Figuren. Numerisch bewertet: Gesamtzahl der gerade möglichen verschiedenen Bewegungen; die Dame zählt doppelt.
- Gefährdung des Computers: Die Zahl der eigenen Figuren, die verlorengingen, wenn der nächsterreichbare, höchstwertige „Stein" des Gegners geschlagen würde.
- Möglichkeit zum Schlagen: Gesamtzahl der gegnerischen Figuren, die gerade geschlagen werden könnten.
- Höchstwertige Figur des Gegners, die gerade geschlagen werden kann. Das wird mit Hilfe einer Identifikationszahl für jede Figur bewertet.

Manchmal – so stellt man nach einigen Spielen fest verhält sich das Schachprogramm völlig widersinnig, und der KIM macht einen recht dummen Zug. Das rührt daher, daß immer die gleiche Strategie angewandt wird, bei der Eröffnung ebenso wie im Schlußspiel, was keinesfalls optimal ist.

In anderen Fällen macht der KIM nur scheinbar zunächst einen widersinnigen Zug, bis sich später herausstellt, wie genial das schließlich war...

Drucker für PET

Für den Heimcomputer PET 2001 von der Firma Commodore wird jetzt ein Drucker angeboten. Er stellt die Zeichen aus einer Matrix von 7 x 8 bzw. 7 x 16 Punkten dar und druckt mit einer Geschwindigkeit von 120 Zeichen/s. Normalerweise finden in einer Zeile 80 Zeichen Platz; bei Sperrschrift, die ebenfalls möglich ist, verringert sich diese Zahl auf 40. Der Drucker kann neben dem üblichen ASCII-Zeichensatz mit Groß- und Kleinschreibung auch sämtliche Grafiksymbole des PET ausgeben. In der Betriebsart „Paging" wird automatisch bei jeder Seite ein oberer und unterer Rand ausgedruckt. Der Datenausgang ist formatiert und unformatiert möglich (links- oder rechtsbündig, Vornullenunterdrückung usw.). Das Gerät wird über den IEC-Bus (IEEE 488) angesteuert.

□ Vertrieb: Elbatex GmbH, Cäcilienstr. 24, 7100 Heilbronn, Tel. (0 71 31) 8 90 01 (oder direkt über Commodore).

Haben Sie schon einmal versucht, im Telefonbuch jemanden ausfindig zu machen, von dem Sie zwar Vornamen und Adresse, nicht aber den Familiennamen wissen? Sicher nicht, denn als intelligenter Mensch wissen Sie, daß das von vornherein zum Scheitern verurteilt wäre. Das Telefonbuch ist eben nur nach einem einzigen Kriterium geordnet, nämlich nach der alphabetischen Folge von Nachnamen. Computer sind da besser dran...

KIM als Nachschlagewerk

Natürlich ist es nicht möglich, das Telefonbuch auch nur einer einzigen Stadt im KIM zu speichern – aber es gibt Hunderte vergleichbarer Beispiele! Denken Sie nur einmal daran, wie schwierig es ist, aus dem Inhaltsverzeichnis des ELEKTRONIK-Sonderheftes II, „Mikrocomputer-Software", herauszufinden, welche Artikel sich mit welchem Prozessortyp befassen; auch dieses Inhaltsverzeichnis kann nur nach einem einzigen Kriterium aufgestellt sein, hier nach Sachgebieten, aber eben nicht nach Prozessortypen.

Ein Computer, zur Not auch ein Mikrocomputer wie der KIM-1, ist hier besser dran: Er läßt sich so programmieren, daß er aus einem Verzeichnis von Daten und Worten alles „ausspuckt", was ein über ein Terminal eingegebenes Wort oder auch nur einige Buchstaben enthält.

Das in der *Tabelle* als 6502-Maschinencode aufgelistete Programm tut dies. Es vergleicht maximal 21 ASCII-Zeichen, die von der Tastatur eingelesen werden, mit einer Datentabelle, und druckt denjenigen Tabellenteil aus, der diese Zeichenfolge in irgendeiner Weise enthält. Jeder Tabellenteil, besser gesagt, jeder zusammengehörige Datenabschnitt, wird von zwei Wagenrücklauf-Zeichen (CR = Carriage Return, ASCII 0D) eingegrenzt und darf max. 255 ASCII-Zeichen enthalten.

Nehmen wir an, die Tabelle enthielte kümmerlicherweise nur zwei Namen, Meier und Huber, nebst den dazugehörigen Telefonnummern. Die Speicherbelegung sähe dann so aus:

Adresse	*ASCII-Zeichen*
0201	Wagenrücklauf (0D)
0202...0207	MEIER (+Leerraum)
0208...020C	23456 (Telefonnummer)
020D	Wagenrücklauf (0D)
020E...0213	HUBER (+Leerraum)
0214...0218	54321 (Telefonnummer)
0219	Wagenrücklauf (0D)
021A	NUL (00, Tabellenende)

Gibt man über die Tastatur MEIER ein, gefolgt von der Return-Taste, so findet das Programm das Wort MEIER und druckt die Zellen 0202...020C aus. Das gleiche geschieht, wenn 23456 eingetastet wird, oder auch nur IER; denn auch diese drei Buchstaben sind in jenem Tabellenteil enthalten. Gibt man dagegen z. B. nur noch ER ein, so stellt der Computer fest, daß diese Zeichenfolge ja auch in HUBER enthalten ist, und druckt konsequent beide Namen nebst Telefonnummern aus.

Es könnte nun der Eindruck entstehen, daß es recht umständlich ist, die Tabelle zu programmieren. Damit dies nicht so ist, hilft das Programm auch beim Einschreiben der Worte und Daten. Das Schreiben der Tabelle geht folgendermaßen vor sich:

Zunächst einmal lädt man mit Hilfe des normalen KIM-Monitor-Programms die Daten 00 in die Zelle 0201. Damit weiß der Computer später, daß er hier mit dem Schreiben der Tabelle beginnen darf.

Dann startet man das Programm an der Adresse 0077 und drückt die „Escape"-Taste am Terminal. So wird in den Schreib-Modus umgeschaltet. Die Tabelle muß mit einem Wagenrücklauf (CR) begonnen werden, und jeder Tabellenteil muß auch wieder mit diesem ASCII-Zeichen (sedez. 0D) abgeschlossen werden. Innerhalb eines Tabellenteils ist kein Wagenrücklauf zulässig!

Ist man mit dem Schreiben der Tabelle fertig, so darf man nicht vergessen, noch einige Male ein NUL-Zeichen auszugeben (die meisten Terminals liefern dieses Zeichen, sedez. 00, u. a. mit der Tastenkombination Control-1). Wenn man später die Tabelle ergänzt, weiß dann der Prozessor gleich, wo er weiterschreiben darf; bei solchen späteren Erweiterungen kann auf das erste CR-Zeichen verzichtet werden, da es ja schon ge-

Notwendiges System: KIM-1 mit TTY oder Terminal. Speicherbedarf: Page 0 für das Programm und Adressen 0201...03FF (oder weiter) für die Datentabelle.

Tabelle. Maschinencode-Programm

Startadresse: 0077

Adresse	Code	Kommentar
0000	A9 00	
0002	85 FA	
0004	A9 02	Tabellenzeiger
0006	85 FB	auf 0200 setzen
0008	A2 17	
000A	A9 00	
000C	95 DF	Buffer löschen
000E	CA	
000F	D0 FB	
0011	20 5A 1E	ASCII-Zeichen holen
0014	C9 1B	„ESC"?
0016	D0 18	nein
0018	C8	ja; Y:=0
0019	20 63 1F	Tab.-Zeiger inkr.
001C	B1 FA	Tabelle abfragen
001E	D0 F9	Freier Raum?
0020	20 5A 1E	ja, Zeichen holen
0023	C8	Y:=0
0024	C9 1B	„ESC"?
0026	F0 D8	ja
0028	91 FA	nein, Zeichen speichern
002A	20 63 1F	Tab.-Zeiger erhöhen...
002D	4C 20 00	...und alles nochmal
0030	C9 0D	„Return"?
0032	F0 07	ja, Wort fertig
0034	95 E0	nein, Zeichen in den Buffer
0036	E8	Bufferindex X erhöhen
0037	E0 15	X=15?
0039	D0 D6	nein, nächstes Zeichen
003B	EA	ja, das muß genügen!
003C	EA	
003D	20 63 1F	Tabelle nach „Return"-
0040	A0 00	Zeichen absuchen,
0042	B1 FA	dabei Y=0
0044	F0 31	
0046	C9 0D	Gefunden?
0048	D0 F3	nein, weitersuchen
004A	A2 00	ja, Zeichen in der
004C	C8	Tabelle mit dem Buffer
004D	B5 E0	vergleichen
004F	F0 0F	Buffer leer, fertig
0051	B1 FA	
0053	C9 0D	„Return"?
0055	F0 E4	ja, nächsten Tab.-Teil suchen
0057	D5 E0	nein, Tab. und Buffer vergl.
0059	D0 EF	
005B	E8	Bufferindex X erhöhen...
005C	D0 EE	...und weitersuchen
005E	EA	
005F	EA	
0060	20 2F 1E	neue Zeile
0063	A0 01	
0065	B1 FA	Zeichen von der Tabelle
0067	F0 0E	laden; fertig, falls Null
0069	C9 0D	„Return"?
006B	F0 CE	ja, Ende dieses Tab.-Teils
006D	84 F9	Y retten...
006F	20 A0 1E	...und Zeichen drucken
0072	A4 F9	Y rückspeichern...
0074	C8	...und um 1 erhöhen
0075	D0 EE	neues Zeichen laden
0077	20 2F 1E	neue Zeile
007A	A9 3F	Fragezeichen
007C	20 A0 1E	ausdrucken
007F	20 9E 1E	Leerraum hinter „?"
0082	4C 00 00	fertig, zurück zum Start
„ESC" (ASCII 1B)		Umschaltung Schreiben Lesen
„Return" (0D)		a) Lesen: Ende der Worteingabe b) Schreiben: Ende eines Tabellenteils

speichert ist. Mit der „Escape"-Taste schaltet man schließlich wieder in den Lese-Modus zurück, und die Tabellendaten sind abrufbereit.

Der „nackte" KIM erlaubt natürlich nur die Ausnutzung des Adressenbereichs 0200...03FF; das Programm selbst begrenzt aber den ausnutzbaren Adressenraum nach oben nicht, so daß man mit einer Speicherexpansion von einigen KByte schon eine ganz hübsche Datenbank zusammenstellen kann.

Der Adressenbereich der „Page 1" (0100...01FF) wurde freigelassen; in ihm läßt sich, um eine möglichst schnelle Bandaufnahme zu ermöglichen, z. B. das „Hypertape"-Programm (s. First Book of KIM) unterbringen.

Der Verfasser speichert so z. B. Telefonnummern, TTL- und CMOS-Typenübersichten, Zeitschriften-Inhaltsverzeichnisse – vielleicht finden Sie noch nützlichere Anwendungen dieses „Nachschlage-Programms".

Herwig Feichtinger

Apple im neuen Gewand

Hinter dem neuen Heimcomputer der Firma ITT verbirgt sich nichts anderes als der bekannte Apple, der in Zukunft von ITT unter eigener Marke verkauft wird. Gleichzeitig wurde er an die europäischen Sicherheitsbestimmungen angepaßt. Es handelt sich um ein Gerät, das mit Hilfe eines Farbfernsehempfängers sogar Farbgrafiken ausgeben kann. Die Auflösung beträgt dabei entweder 40 x 48 oder 40 x 40 Punkte mit vier Zeilen Text. Beschränkt man sich auf die Farben Schwarz, Weiß, Violett und Grün, kann mit einer hochauflösenden Grafik (280 x 192 oder 280 x 160 Punkte mit vier Zeilen Text) gearbeitet werden. Für diese Betriebsart ist mindestens ein Speicher von 12 K RAM erforderlich. Die RAM-Kapazität kann insgesamt bis 48 KByte erweitert werden. Fest implementiert sind ein BASIC-Interpreter und ein Monitor-Programm. Für spätere Erweiterungen sind zwei PROM-Sockel vorhanden. Neben dem Kassettenrecorder können auch ein Drucker und ein Floppy-Disk-Laufwerk angeschlossen werden. Außerdem ist eine Spracheingabekarte erhältlich. Ein Grundgerät mit 16 KByte RAM soll für etwa 4500 DM verkauft werden.

□ Vertrieb: ITT Schaub-Lorenz Vertriebsgesellschaft mbH, Video-Systeme, Postfach 17 20, 7530 Pforzheim, Tel. (0 72 31) 59 23 91.





µP
Technik der Welt
Mikroprozessoren - Mikrorechner
Erlernen Sie die Technik
von morgen mit dem
ITT MIKRO-PROZESSOR LEHRSYSTEM
Die durchdachte Konzeption (zum Patent eingereicht) bietet über Zusatzausrüstungen den nahtlosen Übergang von der Lernphase in den praktischen Anwendungsbereich (also ein „NICHT-NUR-LEHRSYSTEM!")
studio recorder 65
MICROPROZESSOR 8080
ADRESSE
INHALT
NEUER INHALT
EXTENSION-BOX
○ Basis-System: ITT MP-Experimenter mit Stromversorgung, Ein- und Ausgabeeinrichtung, Steckleisten für Systemerweiterungen und 5 Lehrheften (in deutscher, englischer oder französischer Sprache) DM 1.248,- incl. MwSt.
○ Hexadezimal-Eingabe/Anzeige: Erlaubt eine schnelle Eingabe von umfangreichen Programmen DM 198,- incl. MwSt.
○ Extension-Box: In Verbindung mit einem handelsüblichen Cassetten-Recorder können Daten abgespeichert und wieder eingelesen werden. Weitere Funktionen: RAM-Erweiterung, Verschieberoutine für Programmblöcke, zusätzliche I/O-Ports, Benutzung eines USER-PROMs DM 348,- incl. MwSt.
○ PSEUDO-PROM mit Keyboard-Loader: 1 K batteriegepufferte CMOS-Anordnung, die in einem Prozessor-System die Eigenschaft eines ROMs bzw. RePROMS übernehmen kann. Als Ladeeinrichtung dient der Keyboard-Loader.
1 K-PSEUDO-PROM: DM 1.084,- incl. MwSt.
Keyboard-Loader: DM 1.573,- incl. MwSt.
INFORMATIONSSCHECK Bitte einsenden
Name
Straße
HC 08
Wohnort
ITT Fachlehrgänge · Postfach 1570 · 7530 Pforzheim
Schweiz: ITT Fachlehrgänge · Brandschenkestr. 178 · CH-8027 Zürich
Österreich: ITT Lehrmittel · Schottenfeldgasse 13-15 · A-1070 Wien
Ihr Partner in der Ausbildung
Fachlehrgänge
ITT

Herwig Feichtinger

KIM versteht Pseudo-Befehle

Der nachfolgende Beitrag zeigt, daß es mit recht geringem Software-Aufwand schon möglich ist, den Mikrocomputer KIM-1 dazu zu bewegen, übliche aus der Programmiersprache BASIC gewohnte Worte zu „verstehen" und als Steuerbefehle zu interpretieren. Dies erleichtert den Umgang mit dem System ungemein; die direkte Eingabe von Steuerworten erspart oft eine Unzahl mühsamer Einzeloperationen, z. B. das Speichern von Daten auf eine bestimmte Adresse zur Definition des auf Band aufzuzeichnenden Speicherbereichs, das Verschieben von Programmteilen u. a.

Die hier gezeigte Programmversion decodiert nur Steuerbefehle, stellt also keine Programmiersprache im üblichen Sinne dar, da die Befehle nicht als Programmbefehle gespeichert, sondern sofort ausgeführt werden. Die verwendete Methode, wie die Worte decodiert werden, läßt sich aber selbstverständlich auch bei der Verwirklichung von Compilern z. B. zum Aufruf und Transfer ganzer Programm-Moduln einsetzen.

Bild 1. Flußdiagramm

Bild 1 zeigt, wie die einzelnen Worte decodiert werden. Alle möglichen Steuerbefehle sind als Folge von ASCII-Zeichen in einer Tabelle gespeichert. Das Format dieser Tabelle ist:

FF XXXX ABCDEFG FF

Die Daten FF kennzeichnen dabei den Beginn und das Ende eines Tabellenteils, wobei die Enddaten FF gleichzeitig wieder den Beginn eines neuen Wortes darstellen. XXXX steht hier für eine vierstellige Adresse; zu dieser Adresse springt das Programm, wenn das nachfolgende Wort (hier ABCDEFG) decodiert wurde. Das an dieser Adresse dann stehende Programm bewirkt die Ausführung des gewünschten Steuerbefehls.

In der vorliegenden Programmversion darf jedes Wort maximal aus 10 Buchstaben bestehen; ASCII-Zeichen von 00...20 werden ignoriert. Will man z. B. START eingeben, tippt aber versehentlich STURT ein (keine Reaktion), so braucht man nur z. B. die Leertaste zu drücken (ASCII 20) um das Wort richtig einzugeben. (Das Drücken der Return-Taste ist übrigens nicht erforderlich.)

1 Programmieren in Maschinensprache

Nehmen wir einmal an, wir wollen ein Programm zur Anzeige von FFFFFF auf dem KIM-Display realisieren. Dieses Programm kann z. B. die Form haben:

```
0200   A9 FF
0202   85 F9
0204   85 FA
0206   85 FB
0208   20 1F 1F
020B   4C 00 02
```

Die Befehle sind hier bereits in der richtigen Länge pro Zeile geschrieben, und vorne steht eine vierstellige Adresse. Diese Formatierung geschieht bei Verwendung des Pseudo-Befehles „HEX" automatisch; auch „LIST" liefert dieses Format. Wollen wir obiges Programm eingeben, so tun wir einfach folgendes:

Zunächst einmal starten wir das Decodierprogramm an der Adresse 1200; es erfolgt sofort der Ausdruck „READY". Dann geben wir „NEW" ein; wir wollen ja

ein neues Programm realisieren. Ein Leerraum (Space) wird nach NEW automatisch ausgedruckt, sobald das Wort decodiert wurde; jetzt müssen wir noch eine Kennummer eingeben, z. B. 01, die später bei der Bandaufzeichnung als Identifikation für das Anwenderprogramm dient. Jetzt wird sofort am linken Rand der nächsten Zeile die Adresse 0200 ausgedruckt, und das System wartet nun auf eine Eingabe. (NEW geht automatisch immer zur Adresse 0200, da dies für das KIM-System eine sinnvolle Programm-Startadresse ist.)

Da wir das Programm in sedezimaler (hexadezimaler) Form eingeben müssen, beginnen wir diese Eingabe mit dem Steuerbefehl HEX. Er hat zur Folge, daß sofort in der nächsten Zeile nochmals die Adresse 0200 geschrieben wird; wenn man jetzt die Sedezimal-Daten bzw. -Befehle eingibt (A9FF85F985FA usw.), so erkennt das System selbständig die richtige Befehlslänge und liefert ein Format, wie es bei der vorher dargestellten Programm-Auflistung bereits zu sehen ist.

Haben wir alles eingegeben, so steht in der nächsten Zeile die Adresse 020E. Um den Sedezimal-Modus zu beenden, tippen wir noch FF ein; und wieder wird eine neue Zeile mit der Adresse 020E eröffnet. An diese Stelle schreiben wir einfach END. Daraufhin springt das System sofort zur Startadresse (hier 0200). Um die richtige Eingabe zu kontrollieren, kann LIST verwendet werden. Dieser Befehl listet das Programm zwischen Start- und Endadresse wiederum in richtiger Formatierung (wie oben) auf, druckt READY aus und springt wieder an die Startadresse zurück.

2 Einige Steuerbefehle

Mit dem Befehl SAVE kann das Programm jetzt auf Band aufgenommen werden; dabei wird das Hypertape-Programm verwendet, das auch bereits im „First Book of KIM“ veröffentlicht wurde und eine Geschwindigkeit von etwa 600 Bd ermöglicht. Der Befehl OLD, gefolgt von der gewünschten Identifikations-Kennzahl (hier 01), sorgt für das Wiedereinlesen vom Band. Danach erfolgt allerdings ein Sprung zum KIM-Monitor auf die Adresse 0000 bzw. FFFF (letzteres bedeutet eine Fehlermeldung). An der Adresse 1200 kann das Programm dann wieder gestartet werden.

Will man das zuvor eingegebene Programm starten, so braucht man nur (an der Adresse 0200) RUN einzugeben. Um dieses Programm allerdings wieder anzuhalten, muß ein RST- oder NMI-Interrupt verwendet werden.

In diesem Zusammenhang hat es sich als praktisch erwiesen, die Decodier-Startadresse als NMI-Vektor zu verwenden (17FA=00 und 17FB=12) und eine nicht belegte Taste des Terminals oder Fernschreibers mit der Stop-Taste des KIM zu verbinden. Dadurch kann man ein laufendes Anwenderprogramm jederzeit anhalten, indem man diese Taste drückt, und es erfolgt dann der Ausdruck von READY; das System erwartet die Eingabe neuer Steuerbefehle.

Die übrigen Steuerbefehle und ihre Wirkung gehen aus der *Tabelle* hervor. Jeder richtig decodierte Steuerbefehl, dem ein Argument folgen muß, z. B. die Band-Identifikationszahl oder die gewünschte Adresse, erzeugt hinter dem letzten Buchstaben sofort einen Leerraum, dem das Argument folgen muß. Dieser Leerraum wird auch bei SAVE erzeugt, um anzuzeigen, daß der Befehl richtig decodiert wurde. Nach der Aufzeichnungszeit meldet sich das System mit READY wieder.

3 Mögliche Änderungen

Die in *Bild 2* aufgelistete Programmversion ist für ein KIM-System mit 4 KByte externer RAM-Erweiterung gedacht; daher liegt es im Adressenbereich

Tabelle des Befehlsvorrats

Befehl	Argument	Wirkung
BYE	–	Sprung zum KIM-Monitor auf die aktuelle Adresse
RUN	–	Ausführung des Anwenderprogramms ab der aktuellen Adresse
NEW	Progr.-Kenn-ziffer	Speicherung der Kennziffer an der Adresse 17F9 und Sprung an die Adresse 0200
LINE	Neue Adresse	Sprung zu einer neuen Adresse
START	–	Aktuelle Adresse wird als Startadresse deklariert
END	–	Aktuelle Adresse wird als Endadresse deklariert
OLD	Progr.-Kenn-ziffer	Laden eines Programms mit der def. Kennziffer, anschließend Sprung zum KIM-Monitor
SAVE	–	Aufzeichnen des Programms zwischen START und END auf Band (Hypertape)
LIST	–	Auflisten des Programms von der aktuellen Adresse bis zu END; Formatierung mit richtiger Befehlslänge
STRING	ASCII-Zeichen	Speichern von ASCII-Zeichen ab der aktuellen Adresse bis „Escape“ (1B); letzteres wird nicht mitgespeichert
INSERT	n	Einfügen von n Bytes an der aktuellen Adresse und entsprechende Korrektur der Endadresse ($0 \leq n \leq FF$)
DELETE	n	Löschen von n Bytes, sonst wie INSERT
HEX	Sedez.-Bytes	Eingabe eines Programms in Maschinensprache, beendet mit FF; automatische Formatierung wie bei LIST
?	–	Ausdruck bzw. Anzeige des Bytes an der aktuellen Adresse

1100...13FF. Wenn man bei allen 3-Byte-Befehlen, bei denen das dritte Byte 11, 12 oder 13 lautet, eine entsprechende Änderung vornimmt, läßt es sich in beliebige andere „Pages“ (Speicher-Seiten) transferieren. Will man den Befehlsvorrat erweitern, so muß man einige Programmteile aus der Page 13 in einen anderen Speicherteil legen und die Adressen entsprechend korrigieren. Damit wird in dieser Page 13 mehr Platz für die Wortetabelle frei, die nach dem oben erwähnten Format beliebig erweitert werden kann.

```
1100  A9 AD
1102  8D EC 17
1105  20 32 19
1108  A9 27
110A  85 F5
110C  A9 BF
110E  8D 43 17
1111  A2 64
1113  A9 16
1115  20 61 11
1118  A9 2A
111A  20 88 11
111D  AD F9 17
1120  20 70 11
1123  AD F5 17
1126  20 6D 11
1129  AD F6 17
112C  20 6D 11
112F  20 EC 17
1132  20 6D 11
1135  20 EA 19
1138  AD ED 17
113B  CD F7 17
113E  AD EE 17
1141  ED F8 17
1144  90 E9
1146  A9 2F
1148  20 88 11
114B  AD E7 17
114E  20 70 11
1151  AD E8 17
1154  20 70 11
1157  A2 02
1159  A9 04
115B  20 61 11
115E  4C F9 11
1161  86 F1
1163  48
1164  20 88 11
1167  68
1168  C6 F1
116A  D0 F7
116C  60
116D  20 4C 19
1170  48
1171  4A
1172  4A
1173  4A
1174  4A
1175  20 7D 11
1178  68
1179  20 7D 11
117C  60
117D  29 0F
117F  C9 0A
1181  18
1182  30 02
1184  69 07
1186  69 30
1188  A0 07
118A  84 F2
118C  A0 02
118E  84 F3
1190  BE BE 11
1193  48
1194  20 47 17
1197  10 FB
1199  E9 BF 11
119C  8D 44 17
119F  A5 F5
11A1  49 80
11A3  8D 42 17
11A6  85 F5
11A8  CA
11A9  D0 E9
11AB  68
11AC  C6 F3
11AE  F0 05
11B0  30 07
11B2  4A
11B3  90 D8
11B5  A0 00
11B7  F0 D7
11B9  C6 F2
11BB  10 CF
11BD  60
11BE  02 03
```

```
11C0  03
11C1  7E
11C2  20 9F 12
11C5  8D F9 17
11C8  20 9E 1E
11CB  4C 77 13
11CE  20 2F 1E
11D1  20 1E 1E
11D4  C8
11D5  B1 FA
11D7  20 76 12
11DA  20 9E 1E
11DD  C8
11DE  B1 FA
11E0  20 3B 1E
11E3  20 63 1F
11E6  AD F7 17
11E9  C5 FA
11EB  D0 07
11ED  AD F8 17
11F0  C5 FB
11F2  F0 05
11F4  CA
11F5  D0 E3
11F7  F0 D5
11F9  20 8C 1E
11FC  4C 00 12
11FF  00
1200  A2 07
1202  8D 6E 12
1205  20 A0 1E
1208  CA
1209  10 F7
120B  AD F5 17
120E  85 FA
1210  AD F6 17
1213  85 FB
1215  20 2F 1E
1218  20 1E 1E
121B  20 9E 1E
121E  A2 08
1220  A9 00
1222  95 E2
1224  CA
1225  D0 FB
1227  A2 03
1229  20 5A 1E
122C  C9 21
122E  30 EE
1230  95 E0
1232  E0 0E
1234  10 E8
1236  E8
1237  A9 9E
1239  85 F4
123B  A9 13
123D  85 F5
123F  A0 00
1241  B1 F4
1243  C9 FF
1245  D0 10
1247  A0 03
1249  B1 F4
124B  D9 E0 00
124E  D0 03
1250  C8
1251  D0 F6
1253  C9 FF
1255  F0 06
1257  E6 F4
1259  D0 E4
125B  F0 CC
125D  A0 02
125F  B1 F4
1261  85 EF
1263  88
1264  B1 F4
1266  85 F0
1268  20 9E 1E
126B  4C EF 00
126E  59 44 41
1271  45 52
1273  20 20 20
1276  AA
1277  A0 07
1279  8A
```

```
127A  39 80 12
127D  59 8F 12
1280  F0 03
1282  88
1283  D0 F4
1285  6E 97 12
1288  60
1289  0C 1F 0D
128C  87
128D  1F
128E  FF
128F  03
1290  0C 19 08
1293  00
1294  10 20
1296  03
1297  02 03
1299  03
129A  01 01
129C  02 03
129E  01
129F  20 5A 1E
12A2  20 AC 1F
12A5  D0 F8
12A7  4C A3 1F
12AA  20 9F 12
12AD  85 FB
12AF  20 9F 12
12B2  85 FA
12B4  4C 15 12
12B7  20 9F 12
12BA  8D F9 17
12BD  A9 00
12BF  8D F8 17
12C2  A9 00
12C4  4C 0E 12
12C7  A5 FB
12C9  8D F6 17
12CC  A5 FA
12CE  8D F5 17
12D1  4C 0B 12
12D4  A5 FB
12D6  8D F8 17
12D9  A5 FA
12DB  8D F7 17
12DE  4C 0B 12
12E1  20 5A 1E
12E4  C8
12E5  C9 1B
12E7  F0 08
12E9  91 FA
12EB  20 63 1F
12EE  4C E1 12
12F1  4C 15 12
12F4  20 2F 1E
12F7  20 1E 1E
12FA  20 9E 1E
12FD  20 9F 12
1300  C9 FF
1302  D0 03
1304  4C 15 12
1307  91 FA
1309  20 76 12
130C  20 63 1F
130F  CA
1310  F0 E2
1312  20 9E 1E
1315  20 9F 12
1318  91 FA
131A  4C 0C 13
131D  20 9F 12
1320  AA
1321  AD F7 17
1324  85 EF
1326  AD F8 17
1329  85 F0
132B  38
132C  A5 EF
132E  E9 01
1330  85 EF
1332  B0 02
1334  C6 F0
1336  A0 00
1338  B1 EF
133A  85 ED
133C  A4 ED
```

```
133E  91 EF
1340  A5 EF
1342  C5 FA
1344  D0 E5
1346  A5 F0
1348  C5 FB
134A  D0 DB
134C  18
134D  AD F7 17
1350  65 ED
1352  8D F7 17
1355  90 03
1357  EE F8 17
135A  4C 15 12
135D  20 9F 12
1360  AA
1361  A5 FA
1363  85 EF
1365  A5 FB
1367  85 F0
1369  C6 ED
136B  A4 ED
136D  B1 EF
136F  A0 00
1371  91 EF
1373  E6 EF
1375  D0 02
1377  E6 F0
1379  A5 F0
137B  CD F8 17
137E  D0 E9
1380  A5 EF
1382  CD F7 17
1385  D0 E7
1387  38
1388  E5 ED
138A  8D F7 17
138D  B0 03
138F  CE F8 17
1392  4C 15 12
1395  C8
1396  B1 FA
1398  20 3B 1E
139B  4C 15 12
139E  FF

END OF PROGRAM
TABLE FOLLOWS
(FROM =START
        1200)

  * WORD TABLE *

139E  FF FF 1C 4F
13A2  43 59 45 FF
13A6  1C C8 52 55
13AA  4E FF 11 0C
13AE  53 41 56 45
13B2  FF 11 C2 4F
13B6  4C 44 FF 12
13BA  87 4E 45 57
13BE  FF 12 AA 4C
13C2  49 4E 45 FF
13C6  13 1D 49 4E
13CA  53 45 52 54
13CE  FF 13 5D 44
13D2  45 4C 45 54
13D6  45 FF 12 C7
13DA  53 54 41 52
13DE  54 FF 12 D4
13E2  45 4E 44 FF
13E6  12 F4 43 45
13EA  53 FF 12 E1
13EE  53 54 52 49
13F2  4E 47 FF 11
13F6  CE 4C 49 53
13FA  54 FF 13 95
13FE  3F FF 14 14
```

```
HEX-DUMP
OF PROGRAM

1100  A9 AD 8D EC
1104  17 20 32 19
1108  A9 27 85 F5
110C  A9 BF 8D 43
1110  17 A2 64 A9
1114  16 20 61 11
1118  A9 2A 20 88
111C  11 AD F9 17
1120  20 70 11 AD
1124  F5 17 20 6D
1128  11 AD F6 17
112C  20 6D 11 20
1130  EC 17 20 6D
1134  11 20 EA 19
1138  AD ED 17 CD
113C  F7 17 AD EE
1140  17 ED F8 17
1144  90 E9 A9 2F
1148  20 88 11 AD
114C  E7 17 20 70
1150  11 AD E8 17
1154  20 70 11 A2
1158  02 A9 04 20
115C  61 11 4C F9
1160  11 86 F1 48
1164  20 88 11 68
1168  C6 F1 D0 F7
116C  60 20 4C 19
1170  48 4A 4A 4A
1174  4A 20 7D 11
1178  68 20 7D 11
117C  60 29 0F C9
1180  0A 18 30 02
1184  69 07 69 30
1188  A0 07 84 F2
118C  A0 02 84 F3
1190  BE BE 11 48
1194  20 47 17 10
1198  FB E9 BF 11
119C  8D 44 17 A5
11A0  F5 49 80 8D
11A4  42 17 85 F5
11A8  CA D0 E9 68
11AC  C6 F3 F0 05
11B0  30 07 4A 90
11B4  D8 A0 00 F0
11B8  D7 C6 F2 10
11BC  CF 60 02 03
11C0  03 7E 20 9F
11C4  12 8D F9 17
11C8  20 9E 1E 4C
11CC  77 13 20 2F
11D0  1E 20 1E 1E
11D4  C8 B1 FA 20
11D8  76 12 20 9E
11DC  1E C8 B1 FA
11E0  20 3B 1E 20
11E4  63 1F AD F7
11E8  17 C5 FA D0
11EC  07 AD F8 17
11F0  C5 FB F0 05
11F4  CA D0 E3 F0
11F8  D5 20 8C 1E
11FC  4C 00 12 00
1200  A2 07 BD 6E
1204  12 20 A0 1E
1208  CA 10 F7 AD
120C  F5 17 85 FA
1210  AD F6 17 85
1214  FB 20 2F 1E
1218  20 1E 1E 20
121C  9E 1E A2 08
1220  A9 00 95 E2
1224  CA D0 FB A2
1228  03 20 5A 1E
122C  C9 21 30 EE
1230  95 E0 E0 0E
1234  10 E8 E8 A9
1238  9E 85 F4 A9
123C  13 85 F5 A0
1240  00 B1 F4 C9
1244  FF D0 10 A0
1248  03 B1 F4 D9
124C  E0 00 D0 03
1250  C8 D0 F6 C9
1254  FF F0 06 E6
1258  F4 D0 E4 F0
125C  CC A0 02 B1
1260  F4 85 EF 88
1264  B1 F4 85 F0
1268  20 9E 1E 6C
126C  EF 00 59 44
1270  41 45 52 20
1274  20 20 AA A0
1278  07 8A 39 80
127C  12 59 8F 12
1280  F0 03 88 D0
1284  F4 6E 97 12
1288  60 0C 1F 0D
128C  87 1F FF 03
1290  0C 19 08 00
1294  10 20 03 02
1298  03 03 01 01
129C  02 03 01 20
12A0  5A 1E 20 AC
12A4  1F D0 F8 4C
12A8  A3 1F 20 9F
12AC  12 85 FB 20
12B0  9F 12 85 FA
12B4  4C 15 12 20
12B8  9F 12 8D F9
12BC  17 A9 00 8D
12C0  F8 17 A9 00
12C4  4C 0E 12 A5
12C8  FB 8D F6 17
12CC  A5 FA 8D F5
12D0  17 4C 0B 12
12D4  A5 FB 8D F8
12D8  17 A5 FA 8D
12DC  F7 17 4C 0B
12E0  12 20 5A 1E
12E4  C8 C9 1B F0
12E8  08 91 FA 20
12EC  63 1F 4C E1
12F0  12 4C 15 12
12F4  20 2F 1E 20
12F8  1E 1E 20 9E
12FC  1E 20 9F 12
1300  C9 FF D0 03
1304  4C 15 12 91
1308  FA 20 76 12
130C  20 63 1F CA
1310  F0 E2 20 9E
1314  1E 20 9F 12
1318  91 FA 4C 0C
131C  13 20 9F 12
1320  AA AD F7 17
1324  85 EF AD F8
1328  17 85 F0 38
132C  A5 EF E9 01
1330  85 EF B0 02
1334  C6 F0 A0 00
1338  B1 EF 85 ED
133C  A4 ED 91 EF
1340  A5 EF C5 FA
1344  D0 E5 A5 F0
1348  C5 FB D0 DB
134C  18 AD F7 17
1350  65 ED 8D F7
1354  17 90 03 EE
1358  F8 17 4C 15
135C  12 20 9F 12
1360  AA A5 FA 85
1364  EF A5 FB 85
1368  F0 C6 ED A4
136C  ED B1 EF A0
1370  00 91 EF E6
1374  EF D0 02 E6
1378  F0 A5 F0 CD
137C  F8 17 D0 E9
1380  A5 EF CD F7
1384  17 D0 E7 38
1388  E5 ED 8D F7
138C  17 B0 03 CE
1390  F8 17 4C 15
1394  12 C8 B1 FA
1398  20 3B 1E 4C
139C  15 12 FF FF
```

Bild 2. Programmliste

Will man eine Aufgabe mit einem Computer lösen, so muß man sie in einer Sprache formulieren, die dieser Computer versteht. Da die Formulierung von Programmen in Maschinensprache recht umständlich ist, wurden „höhere" Programmiersprachen entwickelt, die diese Formulierung vereinfachten. Für den kommerziellen Bereich (Verwaltung usw.) sind dies unter anderem FORTRAN, COBOL und PL/1, für den wissenschaftlichen hauptsächlich ALGOL. Anfang der 60er Jahre wurde am College of Dartmouth eine Sprache entwickelt, die vor allem Schülern und Studenten das Erlernen der Computerprogrammierung erleichtern soll. Diese Sprache wurde BASIC ***(Beginner's All-purpose Symbolic Instruction Code)*** genannt. Es wäre aber vollkommen falsch, in ihr nur eine Spielerei zu sehen. Sie ist durchaus in der Lage, auch komplexe Probleme zu lösen. Im Hobbycomputer-Bereich ist BASIC praktisch die einzig verwendete höhere Programmiersprache.

Dipl.-Ing. E. Flögel

Einführung in das Programmieren mit BASIC

1 Grundsätzliches

1.1 Compiler und Interpreter

Hier soll gleich auf einen wesentlichen Unterschied zwischen den vorher erwähnten Sprachen FORTRAN, ALGOL usw. und BASIC hingewiesen werden. Ein in ALGOL geschriebenes Programm wird als Ganzes durch den Compiler in ein für die Maschine ausführbares Programm übersetzt. Dieser Compiler ist ebenfalls ein Programm, welches das zu übersetzende Programm auf Fehler (Schreibfehler, Syntaxfehler) untersucht und, falls keine Fehler vorliegen, dieses als Ganzes in die Maschinensprache übersetzt.

Bei BASIC ist dies in der Regel anders. Hier übernimmt ein Interpreter die Ausführung des Programms (es gibt aber auch BASIC-Compiler). Dieser Interpreter, ebenso wie der Compiler ein Programm, überwacht beim Schreiben das Auftreten von Schreibfehlern und meldet diese sofort. Soll das Programm ausgeführt werden, wird es nicht als Ganzes übersetzt und ausgeführt, sondern es wird die erste Zeile des Programms geholt, übersetzt, und die darin enthaltenen Anweisungen werden erledigt. Erst nachdem dies geschehen ist, wird die nächste Zeile behandelt usw., bis die letzte Zeile des Programms übersetzt *und* berechnet ist.

Bei BASIC müssen also gleichzeitig der Interpreter und das Programm im Speicher des Computers vorhanden sein, wogegen bei Programmen, die durch einen Compiler übersetzt werden, man nach der Übersetzung den Compiler löschen und dem aktiven Programm nun den ganzen Speicherraum zur Verfügung stellen kann. Diese Tatsache aber, daß Interpreter und BASIC-Programm gleichzeitig Speicherplatz benötigen, der unter Umständen sehr teuer ist, hat zu der Entwicklung von verschiedenen „Dialekten" dieser Sprache beigetragen.

Für die folgende Einführung soll aber nicht ein nur wenig Speicherplatz benötigendes und nur eingeschränkte Möglichkeiten bietendes „Tiny BASIC" verwendet werden, sondern ein BASIC mit Fließkommaarithmetik, Zeichenkettenverarbeitung und einigen mathematischen Funktionen.

1.2 Programm: eine Folge von Anweisungen

Im weiteren soll nicht nur eine Einführung in die spezielle Sprache BASIC, sondern in das Programmieren überhaupt gegeben werden. Programmieren eines Computers bedeutet ja, ein Problem so in einzelne Schritte zu zerlegen, daß sämtliche auftretenden Möglichkeiten erfaßt und berücksichtigt werden können. Wenn einem Bürocomputer gesagt wird, daß ein Guthaben nur dann vorliegt, wenn der Betrag größer als 0.00 DM ist, und dieser dann bei 0.00 DM eine Mahnung ausschreibt, so irrt hier nicht der Computer, sondern der Programmierer hat einen Fehler gemacht.

Für denjenigen, der einen Computer als Hobby betreiben will, wird wahrscheinlich der größere Reiz darin liegen, eine Aufgabe durch ein Programm zu lösen, als fertige Programme auszuführen.

Ein Programm besteht nun im allgemeinen aus einer Folge von Anweisungen. Form und Schreibweise sind festgelegt, und hierbei auftretende Fehler werden sofort erkannt. Beim Programmieren in BASIC gibt es drei Arten von Anweisungen:

1. Anweisungen, die den Interpreter betreffen:
   z. B. LIST   Diese Anweisung veranlaßt das Aus-

schreiben des vorliegenden Programms.

z. B. RUN — Hiermit wird die Ausführung eines Programms veranlaßt.

2. Anweisungen im Programm:

z. B. GOTO — Sprunganweisung an eine noch festzulegende Stelle im Programm.

z. B. END — Mit dieser Anweisung wird ein Programm beendet.

3. Eingabe/Ausgabe-Anweisungen:

z. B. PRINT — Print veranlaßt die Ausgabe einer Zahl oder eines Zeichens.

z. B. INPUT — Der Computer erwartet die Eingabe einer Zahl oder eines Zeichens.

Das Computersystem wird im allgemeinen aus folgenden Teilen bestehen: Computer mit Eingabetastenfeld, Bildschirm (auch Fernsehgerät) zur Ausgabe und Kassettenrecorder zur Programmspeicherung. Alle drei Komponenten können getrennt aufgestellt werden (z. B. TRS 80, APPLE II) oder in einem Gehäuse vereint sein (z. B. PET).

1.3 Direkte und indirekte Programmierung

Wir wollen nun die Programmierung mit einfachen Beispiel beginnen. Wenn wir mit einem Taschenrechner die Zahlen 3 und 4 addieren wollen, so müssen wir nacheinander die Tasten

(3) (+) (4) (=) betätigen,

das Ergebnis erscheint auf der Anzeige. Lösen wir diese Aufgabe mit unserem Computer, so müssen wir die Ausgabeanweisung PRINT verwenden. Wir schreiben:

```
PRINT 3 + 4
```

Betätigen wir jetzt die RETURN(RET)-Taste, führt der Computer die Anweisung aus, und das Ergebnis erscheint auf dem Bildschirm. Die RETURN-Taste muß nach allen Anweisungen betätigt werden, denn sie zeigt dem Interpreter an, daß die Eingabe beendet ist und er die Anweisung ausführen kann. Wir hätten unser Programm auch durch folgende Eingabe anders ausführen können:

```
LET A = 3        (RET)
LET B = 4        (RET)
LET C=A+B        (RET)
PRINT C          (RET)
```

Hier wird den Variablen A der Wert 3, B der Wert 4 und C der Wert A + B zugewiesen. Die PRINT-Anweisung veranlaßt die Ausgabe des Wertes von C auf den Bildschirm.

Bei dieser Art der Programmierung haben wir den Computer in der „direkten“ Programmierung betrieben. Die weitaus häufigere Art ist die „indirekte“ Programmierung.

Jede indirekte Programmanweisung beginnt mit einer Zeilennummer. Dies ist eine Zahl zwischen 0 und 65 536 (bei einigen anderen Interpretern auch nur zwischen 0 und 9999). Jede Anweisung wird mit dieser Zeilennummer gespeichert, aber noch nicht ausgeführt. Erst die RUN-Anweisung an den Interpreter führt dann zur Ausführung des Programms. Unser Beispiel hätte dann folgende Form:

```
   NEW              (RET) Anweisung an den
                          Interpreter, alle
                          alten Programm-
                          teile zu löschen
10 LET   A = 3      (RET)
20 LET   B = 4      (RET)
30 LET   C = A + B  (RET)
40 PRINT C          (RET)
50 END              (RET) Ende des Programms
   RUN              (RET) Diese Anweisung
                          führt zur Bearbei-
                          tung des Pro-
                          gramms und Aus-
                          gabe von C auf den
                          Bildschirm
```

1.4 Variablennamen

In diesem Beispiel wurden die Variablen A, B, C verwendet. BASIC-Interpreter unterscheiden sich nun oft in der Art der zulässigen Vaiablennamen. Einige lassen nur die Buchstaben A bis Z, andere von A0 und A9 bis Z0 und Z9 zu. Im allgemeinen wird eine Variable durch einen Buchstaben gefolgt von bis zu vier Buchstaben oder Zahlen gekennzeichnet.
Beispiele: AN1, EMMA, NDR1.

Durch solche Variablennamen werden Gleitkommazahlen dargestellt. Nicht verwendet werden dürfen Namen, die als Programm- oder Interpreter-Anweisungen auftreten, wie z. B. RUN, LIST, LET usw.

Daneben gibt es auch die Möglichkeit, Variablennamen für ganze Zahlen (–32 768 ... +32 767) einzuführen. Eine ganze Zahl wird von einer Gleitkommazahl dadurch unterschieden, daß der Variablenname mit einem Prozentzeichen (%) endet. Beispiel: B%, GOOD%.

Außer diesen beiden numerischen Variablen kann auch eine Reihe von Zeichen (Zeichenkette, *String*) als eine Variable aufgefaßt werden. Der Zeichenkette:

DIES IST EIN BEISPIEL

kann auf folgende Weise ein Name zugeordnet werden (siehe 3):

```
10 LET ZK$ = "DIES IST EIN BEISPIEL"
```

ZK ist der Variablenname, das Dollar-Zeichen am Ende des Namens weist darauf hin, daß diese Variable eine Zeichenkette ist. Folgt irgendwo im Programm nun die Anweisung:

```
90 PRINT ZK$
```

so wird auf dem Bildschirm

DIES IST EIN BEISPIEL

ausgeschrieben.

Das gleiche Ergebnis wäre auch mit der Anweisung

```
90 PRINT "DIES IST EIN BEISPIEL"
```

erzielt worden.

Auf die Möglichkeit, Zeichenketten zu verknüpfen und zu verarbeiten, soll später eingegangen werden. In *Tabelle 1* sind die drei Möglichkeiten, Variable zu kennzeichnen, zusammengestellt.

2 BASIC-Anweisungen

2.1 LET und PRINT

Mit den beiden Anweisungen LET und PRINT und den arithmetischen Operationen + für Addition, – für Subtraktion, * für Multiplikation und / für Division läßt sich schon ein Programm schreiben. Als Beispiel wollen wir die Fläche und den Umfang eines Kreises mit dem Radius = 1,5 cm berechnen (*Bild 1*). Als nächstes wollen wir ein Programm betrachten, das uns die Quadratzahlen von 1...10 berechnet (*Bild 2*). Das Programm ist aber in dieser Form umständlich und mit viel Schreibaufwand verbunden. Es läßt sich auf zwei verschiedene Arten wesentlich vereinfachen. Dies ist einmal die bedingte Abfrage mit IF...GOTO, zum zweiten die Schleifenbildung mit FOR...NEXT.

2.2 IF...GOTO

Betrachten wir zunächst ein Programm mit der bedingten Abfrage. Es sollen die Quadratzahlen 1 bis 10 berechnet werden. Wir führen einen Parameter N ein, der nacheinander die Werte 1 bis 10 annimmt; sobald er größer als 10 ist, wird das Programm beendet (*Bild 3*). In diesem Programm wird erst N gleich 1 und N und N^2 ausgedruckt, danach wird N um eins erhöht. Als nächstes folgt die Abfrage, ob N kleiner oder gleich 10 ist. Solange diese Abfrage erfüllt ist (N ist kleiner oder gleich 10), wird zur Zeile 20 zurückgesprungen, wenn nicht (N = 11), wird diese Anweisung übergangen und die nächste Anweisung (Zeile 50 END) ausgeführt.

Die bedingte Abfrage hat also folgende Form:

IF < Ausdruck 1 > < Vergleichsoperator >
< Ausdruck 2 > GOTO < ZNR >

```
]LIST

 10 PRINT "KREISBERECHNUNG"
 20 LET PI=3.14159
 30 LET R=1.5
 40 LET U=2*R*PI
 50 LET F=PI*R*R
 60 PRINT R,U,F
 70 END
]RUN
KREISBERECHNUNG
 1.5              9.42477016     7.06857751
```

Bild 1. Kreisberechnung

```
]LIST

 10 PRINT 1,1*1
 20 PRINT 2,2*2
 30 PRINT 3,3*3
 40 PRINT 4,4*4
 50 PRINT 5,5*5
 60 PRINT 6,6*6
 70 PRINT 7,7*7
 80 PRINT 8,8*8
 90 PRINT 9,9*9
 100 PRINT 10,10*10
 110 END
```

Bild 2. Quadratzahlen von 1...10

```
]LIST

 10 LET N=1
 20 PRINT N,N*N
 30 LET N=N+1
 40 IF N<=10 GOTO 20
 50 END
]RUN
 1             1
 2             4
 3             9
 4             16
 5             25
 6             36
 7             49
 8             64
 9             81
 10            100
```

Bild 3: Quadratzahlen von 1...10 mit Schleifenbildung durch IF...GOTO

Tabelle 1. Variablenbezeichnungen

Art	Anhang an Name	Beispiel
Gleitkommazahl	keiner	DORA, A11, ALPHA
Ganze Zahl	%	GZ%, NO%, A%
Zeichenkette	$	ZK$, MONI$

Ausdruck 1 und 2 können auch zusammengesetzte arithmetische Terme sein, die möglichen Bedingungen und Vergleichsoperatoren sind in *Tabelle 2* zusammengefaßt.

Bevor wir auf die Schleifenbildung mit FOR...NEXT eingehen, noch eine Bemerkung zur Programmerstellung. Es dürfte aufgefallen sein, daß die Zeilennummern nicht fortlaufend mit 1, 2, 3 usw. bezeichnet werden, sondern mit 10, 20, 30 usw. Dies bietet die Möglichkeit, etwa vergessene Zeilen einzufügen. Dies ist in BASIC besonders einfach. Man kann die vergessene Zeile einfach mit der entsprechenden Zeilennummer an das Ende des Programms schreiben, der Interpreter fügt dann diese Zeile an der richtigen Stelle im Programm ein. Sollte im Programm von Bild 3 auch noch die Kubikzahl ausgedruckt werden, so hätten wir am Ende des Programms die Zeile

25 PRINT N * N * N

geschrieben. Lassen wir uns das Programm nun mit LIST ausschreiben, sehen wir, daß diese Zeile zwischen Zeile 20 und 30 eingefügt wurde.

2.3 FOR...NEXT

Eine weitere Möglichkeit, das Programm von Bild 2 zu vereinfachen, ist die Bildung einer Programmschleife mit FOR...NEXT. Die Variable N ist nun Laufparameter in einer Schleife. Bei jedem Durchlaufen der Schleife wird dieser Parameter automatisch um eins erhöht (*Bild 4*). Die Schleife beginnt in Zeile 10. Hier werden die Grenzen für N (1 und 10) festgelegt. Sie endet in Zeile 30 mit der Anweisung NEXT N.

Die FOR/NEXT-Anweisung ist aber in dieser Form noch nicht vollständig. Soll auch die Schrittweite variiert werden, so muß dies auch in der Laufanweisung angegeben werden. Diese hat dann die allgemeine Form:

FOR < Ausdruck 1 > = < Ausdruck 2 > TO
< Ausdruck 3 > STEP < Ausdruck 4 >

Beispiele: FOR N = 5 * A TO B * B STEP 2 * A

Natürlich sind auch geschachtelte Schleifen erlaubt. Im folgenden Programm (*Bild 5*) sollen die Zahlen 0...9 in folgender Form ausgegeben werden.

0 1 2 3 4
5 6 7 8 9

Die äußere Schleife beginnt in Zeile 20 und endet in Zeile 80. Die innere Schleife beginnt in Zeile 30 und endet in Zeile 60. Bei geschachtelten Schleifen muß sichergestellt sein, daß innere Schleifen innerhalb der Grenzen von äußeren Schleifen abgeschlossen werden. Richtige und falsche Schleifenbildung sind in

Tabelle 2. Vergleichsoperatoren

Operator	Bedingung	Beispiel	
=	Gleich	IF A = B	GOTO 20
< >	Ungleich	IF A <> B	GOTO 30
<	Kleiner	IF A < B	GOTO 40
>	Größer	IF A > B	GOTO 50
< =	Kleiner Gleich	IF A < = B	GOTO 60
> =	Größer Gleich	IF A > = B	GOTO 70

Bild 6 dargestellt. Wird ein Programm mit falscher Schleifenbildung gestartet, so muß der Interpreter mit einer Fehlermeldung antworten.

In Bild 5 sind noch einige Besonderheiten zu besprechen. Die erste Anweisung in Zeile 10 kann auch entfallen, da alle Variablen, denen kein Wert zugewiesen wird, automatisch auf Null gesetzt werden. Dies kann aber auch zu Fehlern führen, wenn vergessen worden ist, einer Variablen ein Zahlenwort zuzuweisen. Dies wird bei einem Programmdurchlauf nicht als Fehler erkannt. Zahlenmäßige Berechnungen können so zu völlig falschen Ergebnissen führen.

Eine einfache Formatierung kann mit der PRINT-Anweisung durchgeführt werden. In Zeile 40 folgt auf PRINT N ein Strichpunkt. Dies hat zur Folge, daß kein Wagenrücklauf/Zeilenvorschub, sondern nur ein Leerzeichen ausgegeben wird. Auf eine PRINT-Anweisung ohne weitere Angabe folgt dagegen ein Wagenrücklauf/Zeilenvorschub. Mit einer Anweisung der Form PRINT U,V werden zwischen die Zahlenwerte U und V zwölf Leerzeichen eingefügt, nach der Ausgabe von V erfolgt dann WR/ZV.

2.4 INPUT

Die Eingabe von Zahlenwerten erfolgt durch die INPUT-Anweisung. In Bild 1 hätten wir die Zeile 30 durch INPUT R ersetzen können. Der Rechner wartet dann an dieser Stelle, bis vom Tastenfeld eine Eingabe kommt. Diese Eingabe muß wiederum durch die RETURN-Taste abgeschlossen werden. Danach wird das Programm weiter bearbeitet. Das Programm von Bild 1 soll nun so abgeändert werden, daß nach jeder Berechnung eine neue Eingabe erwartet wird. Durch Eingabe eines negativen Radius wird das Programm beendet (*Bild 7*). Erfolgen in einem Programm mehre Eingaben, so ist es angebracht, diese durch vorheriges Ausschreiben der Variablen zu kennzeichen, damit man sicher ist, welche Eingabe gerade vom Rechner erwartet wird. In Programm 6 hätte man dies durch Einfügen von Zeile 25

25 PRINT "R ="

erreichen können. Einige Interpreter erlauben auch eine INPUT-Anweisung der folgenden Form:

30 INPUT "R ="; R

2.5 IF...THEN

Neben der bedingten Sprunganweisung IF...GOTO gibt es noch eine zweite Art der bedingten Anweisung der Form IF...THEN. Sie hat die allgemeine Form:
IF < Ausdruck 1 > < Bedingung > < Ausdruck 2 > THEN < Anweisung >
Beispiele:
IF A < B THEN PRINT "A IST KLEINER ALS B"
IF A * C > = B * B THEN D = O
Natürlich ist auch die Form

IF A = O THEN 220

erlaubt, 220 ist die Zeilennummer, bei der das Programm weitergeführt wird.

Werden mit der INPUT-Anweisung Eingaben in das Programm gemacht, so ist es oft notwendig, diese Eingabe auf ihre Richtigkeit zu überprüfen. In einem Verbraucher-Statistik-Programm soll das Geschlecht des Verbrauchers (M = männlich, W = weiblich) eingegeben werden. Die beiden folgenden Programme in den *Bildern 8* und *9* erfüllen diese Aufgabe. In beiden Programmen wird die Eingabe überwacht. In Bild 8 allerdings nur, ob ein M eingegeben wurde. Eine Eingabe eines beliebigen Zeichens führt auf die Zeile 40. In Bild 9 dagegen werden die Zeilen 100 oder 200 nur dann ausgeführt, wenn ein M oder W eingegeben wird. Alle anderen Zeichen führen über die Zeile 50 zurück zum Anfang des Programms. Als Eingabevariable wurde hier die Zeichen-(String)Variable A$ verwendet, damit in der IF-Anweisung ein direkter

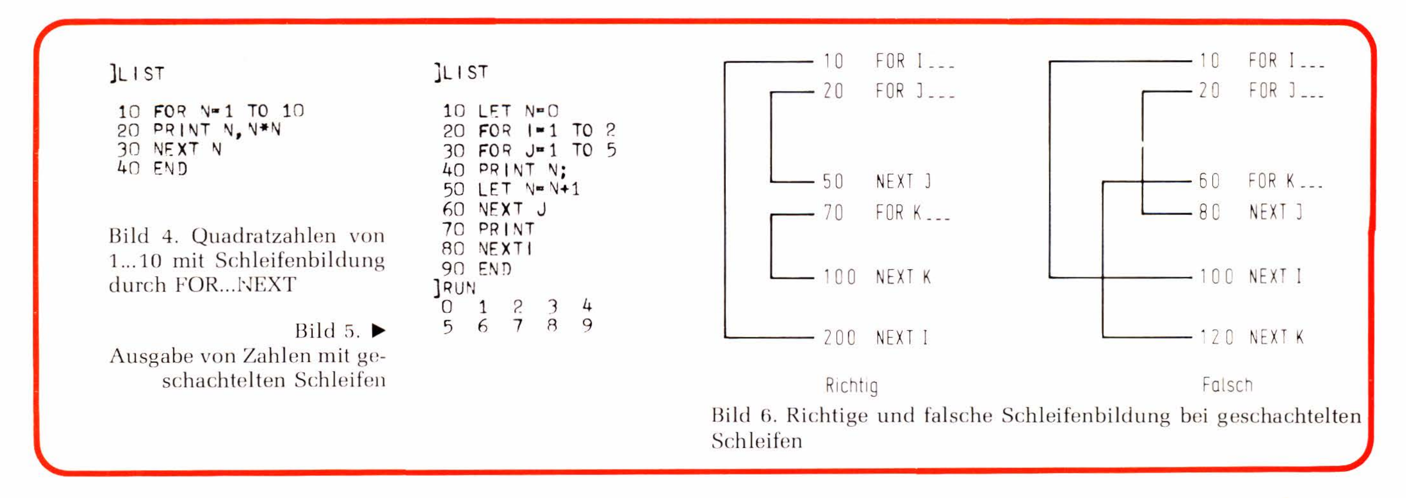


Bild 4. Quadratzahlen von 1...10 mit Schleifenbildung durch FOR...NEXT

Bild 5. ► Ausgabe von Zahlen mit geschachtelten Schleifen

Bild 6. Richtige und falsche Schleifenbildung bei geschachtelten Schleifen

Vergleich mit den ASCII-Zeichen für M oder W durchgeführt werden kann.

2.6 ON...GOTO

Muß ein Programm nach einer Eingabe mehrfach verzweigt werden, kann hierfür ein Verteiler (*Switch*) verwendet werden. Die allgemeine Form eines solchen Verteilers lautet:
ON < Ausdruck > GOTO < ZNR 1 >, < ZNR 2 >, < ZNR 3 >, ... , < ZNR N >
Beispiel: ON K GOTO 150, 200, 250
Ist K = 1, wird nach Zeile 150, mit K = 2 nach Zeile 200, mit K = 3 nach Zeile 250 gesprungen.

Als Programmbeispiel (*Bild 10*) wird die Buchführung für vier Konten gewählt. Die Kontonummern seien 1...4. Eingegeben werden die Kontonummer und der Betrag. Dieser soll positiv sein für Einzahlungen, negativ für Auszahlung.

2.7 DIM

Für eine kleine Zahl von Konten ist diese Art der Programmierung angebracht. Übersteigt aber die Zahl der Konten einen gewissen Betrag, so ist die Schreibarbeit erheblich, das Programm wird zu lang. Abhilfe schafft hier, für die Konten ein Zahlenfeld (*Matrix*) zu vereinbaren. Dieses Zahlenfeld bekommt einen gemeinsamen Namen, z. B. K. Um nun ein Element dieses Zahlenfeldes auszuwählen, wird die Nummer dieses Elementes in Klammern hinter dem Namen angegeben werden. K (5) ist das fünfte Element dieses Feldes, K (1) das erste Element. Vor einem Aufruf eines solchen Elementes muß aber die Länge des Zahlenfeldes festgelegt werden. Dies geschieht mit der DIM-(Dimension)Anweisung. Sie hat die Form:

DIM < Feldname > (< Länge des Feldes >)

Das Programm von Bild 10 soll nun für 100 Konten umgeschrieben werden, wobei noch folgende Vereinbarungen getroffen werden. Wird K = 0 eingegeben, so wird der Kontostand aller Konten ausgedruckt (eine Kontonummer 0 gibt es nicht). Wird eine negative Kontonummer eingegeben, wird nur der Kontostand dieser Nummer ausgegeben, bei positiver Nummer wird der Kontostand nur geändert, aber nicht ausgedruckt. Das Programm (*Bild 11*) wird durch Eingabe von K > 100 beendet.

In diesem Programm wird auch eine in BASIC fest vereinbarte Funktion verwendet. In Zeile 70 wird der Variablen I der *Betrag* der Zahl K zugeordnet. Auf andere noch fest vereinbarte Funktionen wird noch eingegangen werden.

2.8 GOSUB

Wird in einem Programm ein Teilstück häufiger benötigt, so ist es sinnvoll, dieses Teilprogramm als Unterprogramm zu verwenden. Der Sprung in das Unterprogramm erfolgt mit

GOSUB < ZNR >

< ZNR > ist die Zeilennummer, bei der das Unterprogramm beginnt. Das Ende des Unterprogramms wird durch die Anweisung

RETURN

gekennzeichnet. Als Beispiel wählen wir ein Programm, das in einen Text N Leerzeilen einfügt (*Bild 12*). In *Bild 13* ist der Programmablauf schematisch dargestellt. Die GOSUB-Anweisung in Zeile 30 führt zur Ausführung der Zeilen 200...240. Mit der RETURN-Anweisung wird das Programm dann in Zeile 40 fortgesetzt.

Bild 7. Kreisberechnung mit mehrfacher Eingabe

```
]LIST

 10 PRINT "KREISBERECHNUNG"
 20 LET PI=3.14159
 30 INPUT R
 40 IF R<0 GOTO 999
 50 LET U=2*R*PI
 60 LET F=PI*R*R
 70 PRINT R,U,F
 80 GOTO 30
 999 END
]RUN
KREISBERECHNUNG
? 2.1
 2.1            13.1946785     13.8544135
? 4.6
 4.6            28.9026294     66.4760467
? 5.21
 5.21           32.7353611     85.2756318
? -1
```

Bild 8. Überwachung auf richtige Eingabe (einfache Form)

```
]LIST

 10 PRINT "VERBRAUCHER MAENNLICH"
 20 INPUT "ODER WEIBLICH (M ODER W)"; A$
 30 IF A$="M" GOTO 100
 40 PRINT "VERBRAUCHER IST WEIBLICH"
 50 GOTO 999
 100 PRINT "VERBRAUCHER IST MAENNLICH"
 999 END
]
]RUN
VERBRAUCHER MAENNLICH
ODER WEIBLICH (M ODER W)? M
VERBRAUCHER IST MAENNLICH

]RUN
VERBRAUCHER MAENNLICH
ODER WEIBLICH (M ODER W)? S
VERBRAUCHER IST WEIBLICH
```

Bild 9. Überwachung auf richtige Eingabe (bessere Ausführung)

```
]LIST

 10 PRINT "VERBRAUCHER MAENNLICH "
 20 INPUT "ODER WEIBLICH (M ODER W)"; A$
 30 IF A$="M" GOTO 100
 40 IF A$="W" GOTO 200
 50 PRINT "FALSCHE EINGABE"
 60 GOTO 10
 100 PRINT "VERBRAUCHER IST MAENNLICH"
 110 GOTO 999
 200 PRINT "VERBRAUCHER IST WEIBLICH"
 999 END
]RUN
VERBRAUCHER MAENNLICH
ODER WEIBLICH (M ODER W)? W
VERBRAUCHER IST WEIBLICH

]RUN
VERBRAUCHER MAENNLICH
ODER WEIBLICH (M ODER W)? S
FALSCHE EINGABE
VERBRAUCHER MAENNLICH
ODER WEIBLICH (M ODER W)? M
VERBRAUCHER IST MAENNLICH
```

```
]LIST

 10 PRINT "BUCHFUEHRUNG"
 20 INPUT "BETRAG= ";B
 30 INPUT "KONTO NR= ";K
 40 IF K=0 GOTO 200
 50 IF K>4 GOTO 300
 55 IF K<0 GOTO 999
 60 ON K GOTO 70,90,110,130
 70 LET K1=K1+B
 80 GOTO 20
 90 LET K2=K2+B
 100 GOTO 20
 110 LET K3=K3+B
 120 GOTO 20
 130 LET K4=K4+B
 140 GOTO 20
 200 PRINT "DERZEITIGER KONTOSTAND:"
 210 PRINT "KONTO 1";K1;" DM"
 220 PRINT "KONTO 2";K2;" DM"
 230 PRINT "KONTO 3";K3;" DM"
 240 PRINT "KONTO 4";K4;" DM"
 250 GOTO 20
 300 PRINT "UNGUELTIGE KONTONUMMER"
 310 GOTO 20
 999 END
```

Bild 10. Buchführung für wenige Konten. unter Verwendung des Verteilers ON...GOTO

```
 RUN
BUCHFUEHRUNG
BETRAG= ? 1.20
KONTO NR= ? 2
BETRAG= ? 2.50
KONTO NR= ? 3
BETRAG= ? 3.25
KONTO NR= ? 1
BETRAG= ? 4.50
KONTO NR= ? 4
BETRAG= ? 0
KONTO NR= ? 0
DERZEITIGER KONTOSTAND:
KONTO 1 3.25  DM
KONTO 2 1.2  DM
KONTO 3 2.5  DM
KONTO 4 4.5  DM
BETRAG= ? -2.40
KONTO NR= ? 3
BETRAG= ? 0
KONTO NR= ? 0
DERZEITIGER KONTOSTAND:
KONTO 1 3.25  DM
KONTO 2 1.2  DM
KONTO 3 .099999992  DM
KONTO 4 4.5  DM
BETRAG= ? -1
KONTO NR= ? -1
```

```
]LIST

 10 PRINT "BUCHFUEHRUNG"
 20 DIM K    (100)
 30 INPUT "BETRAG= ";B
 40 INPUT "KONTO NR= ";K
 50 IF K=0 GOTO 200
 60 IF K>100 GOTO 300
 70 LET I=ABS(K)
 80 LET K    (I)=K    (I)+B
 90 IF K>0 GOTO 30
 100 PRINT "KONTO NR ";I; K    (I);" DM"
 110 GOTO 30
 200 FOR I=1 TO 100
 210 PRINT "KONTO NR ";I;K    (I);" DM"
 220 NEXT I
 230 GOTO 30
 300 END
```

Bild 11. Buchführung für viele Konten mit der Feldvereinbarung DIM

2.9 Weitere Anweisungen

In BASIC gibt es noch einige andere Anweisungen z. B. READ...DATA, RESTORE usw. Auf diese soll aber nicht weiter eingegangen werden, um den hier gesteckten Rahmen einer Einführung nicht zu sprengen. Es gibt inzwischen genügend Lehrbücher, die dem Interessierten eine weitere Einarbeitung ermöglichen. Auch sollen hier die in BASIC enthaltenen Funktionen nur kurz erwähnt werden. Die Wertzuweisung einer Funktion erfolgt in der Form
< Variable > = < Funktion > (< Argument >)
z. B. X = SIN (A), Y = SQR (4).

Im allgemeinen sind die Winkelfunktionen Sinus, Cosinus, Tangens usw., die Wurzelfunktion SQR (*Square Root*), die Bildung von ganzen Zahlen INT (*Integer*), die Bildung von Zufallszahlen RND (*Random*) usw. in BASIC enthalten. Daneben gibt es noch eine weitere Funktion, der Form X = USR (x) oder CALL (x), wobei das Argument x die Adresse eines internen Maschinenprogramms darstellt, das mit dieser Anweisung zur Ausführung gebracht wird. Die Anwendung dieser Funktion setzt aber Kenntnisse der internen Programmierung im Maschinencode voraus.

3 Zeichenketten (Strings)

Eine Zeichenkette ist eine Reihe von Zeichen (Buchstaben, Zahlen, Sonderzeichen), also einfach ein Text. Wird diesem Text ein Name zugeordnet, so soll dieser als Textvariable (alphanumerische Variable) bezeichnet werden. Zur Kennzeichnung muß mit einem $-Zeichen abgeschlossen werden. Die Zuweisung erfolgt dann auf folgende Weise:

10 LET A$ = "ABCDE"

oder

20 LET OTTO$ = "GUTEN TAG!"

Anfang und Ende einer Zeichenkette werden mit einem Doppelapostroph (") gekennzeichnet.

```
]LIST

 10 INPUT "N= ";N
 20 PRINT "DIESER TEXT"
 30 GOSUB 200
 40 PRINT "WIRD MIT N LEERZEILEN"
 50 GOSUB 200
 60 PRINT "ZWISCHEN DEN ZEILEN"
 70 GOSUB 200
 80 PRINT "AUSGEDRUCKT"
 90 END
 200 IF N>0 GOTO 220
 210 GOTO 250
 220 FOR I=1 TO N
 230 PRINT
 240 NEXT I
 250 RETURN
]
]RUN
N= ? 2
DIESER TEXT


WIRD MIT N LEERZEILEN


ZWISCHEN DEN ZEILEN


AUSGEDRUCKT
```

Bild 12. Einfügen von Leerzeilen in einen Text unter Verwendung des Unterprogrammsprunges GOSUB...RETURN.

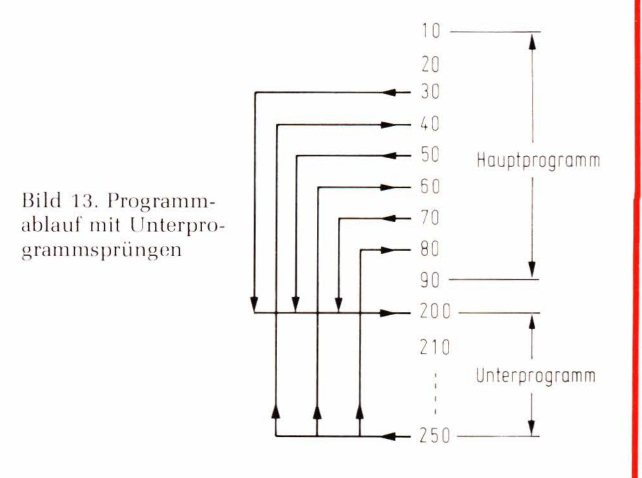


Bild 13. Programmablauf mit Unterprogrammsprüngen

Bei der Verarbeitung dieser Texte gibt es zwei unterschiedliche Versionen. Die eine wurde von der Firma Hewlett-Packard (HP), die andere von der Firma Digital Equipment (DEC) entwickelt. Die weiteren Betrachtungen beziehen sich auf die Fassung von DEC, insbesondere auf die von der Mikrosoft Inc. im Apple-II-Computer verwendete Fassung.

3.1 Die Funktionen LEN, LEFT$, RIGHT$, MID$

An einem einfachen Beispiel soll die Wirkung dieser Funktion auf einen Text gezeigt werden. Es ist die Zeichenkette

B$ = "DIES IST EIN BEISPIEL"

gegeben. Mit der Funktion LEN (< NAME$ >) erhält man die Anzahl der einzelnen Zeichen dieser Kette.

PRINT LEN(B$) ergibt 21

PRINT LEN("JA") ergibt 2.

Die maximale Länge einer Zeichenkette hängt von dem verwendeten Interpreter ab und kann bis zu 255 Zeichen betragen. Eine weitere Möglichkeit der Verarbeitung besteht darin, Teilfolgen (Teiltexte, *Substrings*) zu bilden. Hierzu dienen die Funktionen LEFT$, RIGHT$ und MID$.

PRINT LEFT$(B$, 4) ergibt DIES.

Die Funktion LEFT$ (< NAME$ >, N) bildet eine Teilfolge aus dem Text NAME$, der am linken Rand beginnt und N Zeichen enthält. Ein Beispiel stellt *Bild 14* vor.

Mit RIGHT$ (< NAME$ >, N) wird eine Teilfolge mit N Elementen bis zum rechten Rand des Textes gebildet. Ersetzt man in Bild 14 in Zeile 30 LEFT$ durch RIGHT$, so erhält man den dort angegebenen zweiten Ausdruck.

Soll eine Teilfolge gebildet werden, die innerhalb des Textes beginnt, so verwendet man die Funktion MID$. Diese Funktion kann 2 oder 3 Argumente haben. Die Anweisung MID$ (< NAME$ >, N) ergibt eine Teilfolge des Textes NAME$, die beim Element N beginnt und bis zum rechten Rand des Textes reicht, während die Anweisung MID$ (< NAME$ >, N, K) beim Element N beginnt und K Zeichen lang ist (*Bild 15*).

3.2 Zusammenfügen von Zeichenketten

So wie man mit den angeführten Funktionen einen Text in Teile zerlegen kann, kann man diese Teile wiederum zu einem neuen Text zusammensetzen. Hierzu dient das Pluszeichen (+), mit dem man einfach Teiltexte „addiert". Ein kleines Beispiel zeigt *Bild 16*, in dem die Texte B$ und NA$ zu einem neuen Text B$ + NA$ zusammengefügt werden.

Die DEC-Version von BASIC erlaubt auch die Bildung von Feldern mit Zeichenketten (Textfelder, Listen, String-Arrays). Wie bei Zahlenfeldern muß durch eine DIM-Anweisung die Länge des Feldes festgelegt werden. Ein Element dieses Feldes ist dann eine Zeichenkette mit wiederum bis zu 255 Zeichen. Damit ist es leicht, zum Beispiel Namenslisten zu verarbeiten. *Bild 17* zeigt die Verwendung von solchen Textfeldern. Hier sind mehrere Listen in einem zweidimensionalen Feld vereinbart, aus denen durch Zusammenfügen von zufällig ausgewählten Elementen ein neuer Text gebildet wird.

Für die Erzeugung von Zufallszahlen wird die RND(Random)-Funktion verwendet. Die Anweisung

LET Z = RND (1)

weist der Variablen Z eine Zufallszahl zwischen 0 und 1 zu. Bei einigen BASIC-Interpretern wird das Argument der RND-Funktion übergangen. Bei anderen bildet das Argument die Basis zur Erzeugung neuer Zufallszahlen. Um zu vermeiden, daß bei jedem Neustart eines Programms die gleiche Folge von Zufallszahlen gebildet wird, muß vorher die Anweisung RANDO-

```
]LIST

 10 LET N$="MONIKA"
 20 FOR K=1 TO LEN(N$)
 30 PRINT LEFT$(N$,K)
 40 NEXT K
 50 END
]
]RUN
M
MO
MON
MONI
MONIK
MONIKA
```

```
]
]30 PRINT RIGHT$(N$,K)
]
]RUN
A
KA
IKA
NIKA
ONIKA
MONIKA
```

Bild 14. Bildung von Teilfolgen mit den Funktionen LEN, LEFT$, RIGHT$

```
]LIST

 10 LET K$="ELEKTRONIK"
 20 PRINT MID$(K$,6,4)
 25 PRINT
 30 FOR N=1 TO LEN(K$)
 40 PRINT MID$(K$,N,1)
 50 NEXT N
 60 END
]
]RUN
RONI

E
L
E
K
T
R
O
N
I
K
```

Bild 15. Bildung von Teilfolgen durch die Funktion MID$

```
]LIST

 10 INPUT"NAME ";NA$
 20 LET B$="HALLO "
 30 PRINT B$+NA$
 40 END
]
]RUN
NAME ? KARL
HALLO KARL
```

Bild 16. Zusammenfügen von Zeichenketten

MIZE erfolgen. Durch sie wird sichergestellt, daß bei jedem RUN neue Zahlen auftreten. Andere Interpreter vermeiden dies, indem sie zur Erzeugung einer Zufallszahl den Inhalt interner Speicherzellen verwenden, wobei dieser Inhalt von der Vorgeschichte (Einschaltdauer, Zahl der Tastenbetätigungen, usw.) des Computers abhängt. Zu Bild 17 noch einige Hinweise: Die meisten neueren Interpreter benötigen nicht mehr die LET Anweisung. Anstatt 10 LET A = 5 kann einfach 10 A = 5 geschrieben werden. Desweiteren können mehrere Anweisungen in einer Zeile zusammengefaßt werden. Die Trennung erfolgt je nach Interpreter entweder mit einem Doppelpunkt (:), einem Schrägstrich (/) oder einem Ausrufungszeichen (!). Statt

```
10 A = 5
20 B = 3          schreibt man
10 A = 5:B = 3
```

Der Ablauf des Programms in Bild 17 ist folgender: A$ ist ein zweidimensionales Textfeld mit 7 Reihen und 5 Spalten. Dieses Feld wird mit der READ-Anweisung mit den unter DATA angegebenen Begriffen gefüllt. Der Matrix Z werden acht Zufallszahlen zugewiesen. In den Zeilen 100...140 wird ein Text mit den zufällig ausgewählten Begriffen zusammengesetzt und ausgegeben. Auch die im Text enthaltenen Zahlenwerte werden mit Zufallszahlen berechnet.

3.3 Vergleich von Zeichenketten

Im Computer werden die Zeichen als eine zweistellige Sedezimalzahl dargestellt. Dem Zeichen A entspricht die Zahl 41, dem B entspricht die Zahl 42 usw. Diese Darstellung ermöglicht es, Zeichenketten zu vergleichen. Dabei wird das erste Zeichen des Textes 1 mit dem ersten Zeichen des Textes 2, das zweite Zeichen mit dem zweiten Zeichen des anderen Textes verglichen, solange, bis ein Unterschied (größer oder kleiner) auftritt, oder aber alle Zeichen eines Textes abgearbeitet sind. Beispiele finden sich schon in den Programmen der Bilder 8 und 9. Dort wird die Eingabe A$ mit dem Zeichen "M" bzw. "W" verglichen. Ein anderes Beispiel ist das Programm in *Bild 18*. Hier soll eine Namensliste erstellt werden. Die Namen werden in der Form

< VORNAME > < NACHNAME >

eingegeben. Der Zwischenraum zwischen Vor- und Nachname stellt hier ein wichtiges Zeichen dar, denn er dient als Trennzeichen (*Delimiter*). Die Eingabe erfolgt auf die Zeichenkette N$. In den Zeilen 40...70 wird der letzte Zwischenraum gesucht und die Nummer dieser Stelle der Variablen S zugeordnet (Zeile 60). Ist kein Zwischenraum vorhanden wird die Eingabe beendet. In den Zeilen 90...110 wird der Name in Vorname (N1$) und Nachname (N2$) zerlegt und in ein neues Element der Form

```
]LIST

 10 DIM A$(6,4)
 20 DIM Z(7)
 30 FOR I=0 TO 6
 40 FOR J=0 TO 4
 50 READ A$(I,J)
 60 NEXT J:NEXT I
 70 FOR I=0 TO 7
 80 Z(I)=INT(5*RND(1)):NEXT I
 90 PRINT:PRINT:PRINT
 100 PRINT"UNSER "+A$(0,Z(0))+" "+A$(1,Z(1))+" "+A$(2,Z(2))+" IM ";
 110 PRINTA$(3,Z(3))+"EN "+A$(4,Z(4))+"PULT ERZEUGT"
 120 PRINT A$(2,Z(5))+" GERAEUSCHE,MIT "+A$(5,Z(6))+"EN ";
 130 PRINT INT(1000*RND(1));" "+A$(6,Z(7))+"."
 140 PRINT:PRINT "JETZT NUR NOCH";INT(600*RND(1));".98 DM"
 200 DATA "KL","UV","ZX","MR","BLA"
 210 DATA "RESLY","UGLY","MULTY","ORGY","TINY"
 220 DATA "WAH","BEH","FUZ","ORCH","URGS"
 230 DATA "SCHWARZ","FREUNDLICH","RESISTIV","EINHEITLICH","SUPERWEISS"
 240 DATA "BEDIEN","DREH","KLAVIER","GLITCH","BALANCE"
 250 DATA "HERVORRAGEND","SINUSARTIG","DREIECKIG","UEBERSICHTLICH","STABIL"
 260 DATA "OKTAVEN","PS","WATT","HERTZ","BAUD"
 300 END
]RUN


UNSER KL MULTY FUZ IM RESISTIVEN DREHPULT ERZEUGT
FUZ GERAEUSCHE,MIT DREIECKIGEN  357  WATT.

JETZT NUR NOCH 476 .98 DM

]RUN


UNSER KL MULTY WAH IM RESISTIVEN KLAVIERPULT ERZEUGT
WAH GERAEUSCHE,MIT DREIECKIGEN  374  PS.

JETZT NUR NOCH 208 .98 DM

]RUN

UNSER ZX UGLY FUZ IM SUPERWEISSEN DREHPULT ERZEUGT
FUZ GERAEUSCHE,MIT HERVORRAGENDEN  544  PS.

JETZT NUR NOCH 265 .98 DM
```

Bild 17. Zusammenfügen von Zeichenketten: Erzeugung von Zufallstext

```
]LIST

 10 DIM NAME$(50)
 20 D=D+1
 30 INPUT N$
 40 C=1:L=LEN(N$):S=0
 50 IF C>L GOTO 80
 60 IF MID$(N$,C,1)=" " THEN S=C
 70 C=C+1:GOTO 50
 80 IF S=0 GOTO 200
 90 N1$=LEFT$(N$,S-1)
 100 N2$=RIGHT$(N$,L-S)
 110 NAME$(D)=N2$+","+N1$
 120 GOTO 20
 200 F=0
 210 FOR I=2 TO D-1
 220 IF NAME$(I)>NAME$(I-1) GOTO270
 230 F=1
 240 N$=NAME$(I):NAME$(I)=NAME$(I-1)
 250 NAME$(I-1)=N$
 270 NEXT I
 280 IF F>0 GOTO 200
 290 PRINT:PRINT:PRINT
 300 FORI=1 TO D-1
 310 PRINT NAME$(I)
 320 NEXT I
 330 END
]RUN
? KLAUS MAYER
? MONIKA FLOEGEL
? KATI HEWEL
? RALF ZERLING
? UWE ROSCHER
? RAINER BENZ
? FRANK ZERLING
? AUS


BENZ,RAINER
FLOEGEL,MONIKA
HEWEL,KATI
MAYER,KLAUS
ROSCHER,UWE
ZERLING,FRANK
ZERLING,RALF

]
```

Bild 18: Vergleich von Zeichenketten: Eingabe und Sortieren einer Namensliste

< NACHNAME >, < VORNAME >

umgewandelt. Dieses Element wird im Feld NAME$ gespeichert. Nach Beendigung der Eingabe wird dieses Feld sortiert (Zeile 200...280) und danach ausgegeben.

4 Programmerstellung

Für die Erstellung von Programmen in BASIC gilt genau das gleiche wie für alle anderen Sprachen auch. Vor der Programmierung sollte man, dies gilt besonders für längere Programme, ein Struktur- oder Flußdiagramm aufstellen. Dies leistet dann gleichzeitig gute Dienste bei einer etwaigen Fehlersuche. Ferner sollten längere Programme in kleinere Teilprogramme zerlegt werden, die für sich getestet werden können. Das Programmieren in BASIC hat einen entscheidenden Vorteil gegenüber anderen Sprachen. Es ist ein interaktiver Vorgang zwischen Mensch und Computer. So kann man während des Programmierens an beliebiger Stelle eine END- oder STOP-Anweisung einfügen und durch Starten eines „RUNS“ das Programm bis an diese Stelle überprüfen. Es entfällt also der Gang zum Rechenzentrum, die Abgabe des Programms, das Warten auf die Ausgabe, um festzustellen, daß doch noch ein Fehler vorhanden ist.

Aber auch wenn ein Programm fehlerfrei läuft und schöne Rechenergebnisse liefert, ist noch einige Skepsis angebracht. Numerische Berechnungen sollten wenigstens einmal überschlägig mit dem Taschenrechner geprüft werden. Das vor allem deshalb, da BASIC keine Fehlermeldung liefert, wenn einer Variablen kein Wert zugewiesen wurde, sondern diese einfach als Null annimmt. Ferner empfiehlt es sich auch, mit der REM(Remark)-Anweisung Erklärungen in das Programm einzufügen. Einmal um anderen bei einem Programmaustausch das Programm verständlich zu machen, zum zweiten zur eigenen Gedächtnisunterstützung, damit man nicht später vor der Frage steht: „Was habe ich damals eigentlich *wie* programmiert?“

Bei Hobbycomputern ist es üblich, Programme auf normale Tonkassetten mit einem handelsüblichen Kassettenrecorder zu speichern. Mit der Anweisung SAVE wird das erstellte Programm auf die Kassette geschrieben, mit LOAD wieder in den Computer eingelesen. Will man nun mehrere Programme auf einer Kassette speichern, so sollte man jedem Programm eine Überschrift geben, z. B.

```
5 REM PRIMFAKTORENZERLEGUNG
7 REM FASSUNG VOM 3.5.78
```

Durch die Anweisung LIST 5–7 kann man sich diese Überschrift wieder ausschreiben lassen und ist dann sicher, auch das richtige Programm geladen zu haben.

Auf einen Nachteil von BASIC in der vorliegenden Form bei Hobbycomputern soll noch hingewiesen werden: Mit der SAVE-Anweisung wird nur das Programm, nicht aber die vom Programm erzeugten Daten gespeichert. Im Programm von Bild 18 z. B. werden nur die Zeilen 10...300 gespeichert, nicht aber die eingegebene Namensliste. Dieser Nachteil läßt sich aber durch spezielle, in Maschinensprache geschriebene Programme beheben.

5 Schlußbemerkung

Es liegt wohl am B im Wort BASIC, das für Beginners d. h. Anfänger steht, daß diese Sprache wirklich nur als eine Sprache für Anfänger angesehen wird. In Verbindung mit einem Hobbycomputer lassen sich mit ihr durchaus kommerzielle Probleme direkt vom Schreibtisch aus bearbeiten. Dies sind z. B. einfache Buchführung, Lagerhaltung, Katalogisierung von Privatsammlungen, Literaturverzeichnisse, technische und wissenschaftliche Berechnungen usw. Auch der Einsatz im Labor bei der Meßdatenverarbeitung ist möglich. Die Meßdatenerfassung kann mit schnellen, im Maschinencode geschriebenen Programmen erfolgen, während für die Auswertung dieser Daten das einfach zu programmierende BASIC verwendet wird.

Für viele liegt aber die Schwierigkeit nicht im „Wie programmieren?“, sondern im „Was programmieren?“. Deshalb sind zum Schluß noch einige Programmieraufgaben zum Üben angefügt.

6 Übungsaufgaben

1. Gesucht sind alle dreistelligen Zahlen ABC, für die gilt $$A^3 + B^3 + C^3 = ABC.$$
2. Gesucht sind alle Lösungen der folgenden Addition
   MORE + WIRE = MONEY
3. Umwandlung von ganzzahligen Dezimalzahlen in Zahlen mit vorgebbarer Basis zwischen 2 und 16.
4. Zerlegung einer Zahl in Primfaktoren
5. Umwandlung von Dezimalzahlen in römische Zahlen.
6. Aufstellung des Stichwortverzeichnisses eines Buches. Eingabe des Stichwortes mit Seitenzahl, Einsortieren des Begriffes in das Verzeichnis. Falls dieser Begriff schon vorhanden ist, Einfügen der neuen Seitenzahl, Sortieren der Begriffe, Sortieren der Seitenzahlen.
7. Lernprogramm: Ausgabe eines deutschen Wortes, Eingabe der englischen (französischen, lateinischen usw.) Übersetzung. Prüfung der Schreibweise und Richtigkeit. Angabe der Fehler.

Literatur

|1| *Alteneder, A., Offelder, C.:* BASIC-Praktikum. Lernprogramm. Siemens Aktiengesellschaft, München 1972.

|2| *Alteneder, A., Offelder, C.:* BASIC-Praktikum. Katalog. Siemens Aktiengesellschaft, München 1972.

|3| *Ramp, H.O.:* BASIC-Praxis. Oldenbourg, Wien 1972.

|4| *Schärf, J., Kunesch, A.:* BASIC für Kaufleute. Oldenbourg, Wien 1972.

|5| *Schärf, J.:* BASIC für Anfänger. 3. Aufl., Oldenbourg, Wien 1974.

|6| *Mägerle, E.:* Einführung in das Programmieren in BASIC. De Gruyter, Berlin 1974.

|7| *Rehbein, H.:* BASIC leicht gemacht. Eine Einführung und 50 vollst. Übungsaufgaben. 2. Aufl., VDI-Verlag, Düsseldorf 1974.

|8| *Sack, J., Meadows, J.:* BASIC. Eine Einführung. Science Research Associates, Stuttgart 1976.

|9| *Weber, Türschmann:* BASIC 1. UNI Taschenbücher 588, 1977.

|10| *Weber, Türschmann:* BASIC 2. (Mit vollständiger Literaturübersicht). UNI Taschenbücher 589, 1977.

|11| *Hase, V., Stucky, W.:* BASIC. Programmieren für Anfänger. BI Hochschultaschenbücher Band 744, 1977.

BASIC ≠ BASIC

Der Titel dieses Beitrages mag zunächst verwundern, denn die Programmiersprache BASIC ist ja genormt. Doch die Praxis zeigt, daß gerade bei BASIC-Versionen für Mikrocomputer im Regelfall von dieser Norm abgewichen wird – meist aus Gründen der Vereinfachung. Dies führt oft dazu, daß nach dem Eintippen (z.B. aus einer Zeitschrift) das Programm einfach nicht laufen will. Nun beginnt das Rätselraten: Läuft es von der Konstruktion her nicht – sprich lief es vielleicht noch nie – oder wurde ein Tippfehler gemacht? Oder liegt etwa an irgendeiner Stelle eine Inkompatibilität vor? – in diesem Fall ist eben BASIC nicht gleich BASIC.

Die folgenden Tips sollen dem Leser dabei helfen, solche Inkompatibilitäten zu finden und durch Umschreiben zu beseitigen. Natürlich kann der Beitrag nur als Anregung dienen, da längst nicht alle Fälle geschildert werden können.

Syntaktische Inkompatibilitäten

Darunter werden Inkompatibilitäten verstanden, die nach Starten des Programms auf einem anderen BASIC-System eine Fehlermeldung hervorrufen. Sie sind meist leicht behebbar und vor allem leicht auffindbar. Ein paar Beispiele sollen dem Leser ein Gefühl für derartige Fehler geben:

CHANGE

Der Befehl kann nach zwei verschiedenen Weisen aufgeführt werden. CHANGE ‹stringvar› TO ‹var› oder CHANGE ‹var› TO ‹stringvar›. Er wird zur Umwandlung von Zahlen in einen String (ASCII-Zeichenfolge) und umgekehrt verwendet. Bei den meisten BASIC-Interpretern wird diese Funktion durch zwei gesonderte Befehle erreicht: Mit VAL wird die Umwandlung eines Strings in eine Zahl (falls der String eine Zahl darstellt) und mit dem Befehl STR$ die Umwandlung einer Zahl in einen String erreicht.

INSTR

Dieser Befehl wurde deshalb hier aufgenommen, da er bei den meisten BASIC-Versionen nicht vorhanden ist, andererseits aber sehr gerne verwendet wird. Er bewirkt das Suchen nach einem Teilstring innerhalb eines größeren Strings. Das Format lautet: INSTR (‹var›, ‹stringvar›, ‹stringvar›). Es wird dabei nach dem zweiten String gesucht (innnerhalb des ersten Strings), die Suche beginnt bei dem Zeichen, dessen Position durch den Wert der Variablen (erste Position) bestimmt wird. Als Ergebnis wird die Position des Zeichens angegeben, von dem ab die Suche Erfolg hatte. Der Wert 0 steht für den Fall, daß der Teilstring nirgends auftrat.

LINPUT

Dieser Befehl, der bei manchen BASIC-Versionen als LINE INPUT geschrieben werden muß, besitzt folgendes Format: LINPUT ‹stringvar›. Im Gegensatz zu dem Befehl INPUT kann hier eine ganze Eingabezeile eingegeben werden, die auch Kommas etc. (alle Sonderzeichen) verwenden darf. Die Zeile gilt mit der Eingabe von CR (Wagenrücklauf) als beendet.

RESTORE

Bei diesem Befehl sind in manchen BASIC-Systemen zusätzliche Parameter erlaubt. Normalerweise bewirkt er das Rückstellen eines Datenzeigers, der mit READ gesteuert wird. Das Rückstellen erfolgt immer auf das erste Datenwort, das in der DATA-Anweisung gegeben wurde. Mit RESTORE ‹var›, wie es in manchen BASIC-Versionen erlaubt ist, kann dieser Datenzeiger auf eine bestimmte Zeile gesetzt werden. Ferner gibt es BASIC-Versionen, bei denen zwischen dem Lesen eines Strings und einer Variablen unterschieden wird. Mit RESTORE* kann der numerische Datenzeiger zurückgesetzt werden und mit RESTORE$ der String-Datenzeiger.

PEEK, POKE

Der Befehl PEEK bewirkt das Lesen eines Bytes aus dem Speicher, und mit POKE kann ein Wert in eine Speicherzelle geschrieben werden. Dabei ist zu beachten, daß die Adresse der Speicherzelle absolut angegeben werden muß. Daher können zusätzliche Schwierigkeiten bei der Übernahme eines Programms mit diesen Befehlen entstehen, da vielleicht der betreffende Speicherplatz auf der eigenen bzw. anderen Maschine nicht vorhanden oder anders belegt ist. Die beiden Befehle PEEK und POKE werden manchmal auch anders dargestellt, z.B. mit dem Zeichen ∂. ∂ (1000) = 10 bewirkt die Ablage der Zahl 10 auf der Speicherzelle 1000 *(nicht mit Arrays zu verwechseln).*

USR,CALL

Mit diesen Befehlen ist es ähnlich. Sie bewirken den Aufruf eines Maschinenunterprogramms. Entweder sind sie bei dem eigenen BASIC überhaupt nicht vorhanden oder, wenn sie vorhanden sind, besitzt man vielleicht einen anderen Prozessor. Im ersten Fall kann versucht werden, den Maschinenteil durch ein BASIC-Programm zu ersetzen. Im zweiten Fall muß

der Maschinenteil neu in die eigene Prozessorsprache übersetzt werden.

HEX

Mit dem Befehl HEX kann eine Umwandlung eines sedezimalen Strings in eine dezimale Zahl erreicht werden. HEX(A) entspricht z.B. dem Wert 10. Manchmal werden dafür andere Zeichen verwendet wie # oder &.

MID$

Dieser Befehl bewirkt die Ausgabe eines Teilstrings aus einem String. Die Form ist MID$ (‹string var›, ‹var›, ‹var›). Aus dem angegebenen String werden beginnend bei der Position, die durch die erste Variable angegeben ist, n Zeichen ausgegeben, die durch den zweiten Wert bestimmt sind.

LEFT$, RIGHT$

Diese Befehle sind ähnlich dem MID$-Befehl, der String wird aber linksbündig bzw. rechtsbündig ausgegeben. Format: LEFT$ (‹stringvar›, ‹var›) bzw. RIGHT$ (‹stringvar›, ‹var›).

Semantische Inkompatibilitäten

Hierbei handelt es sich um Fehler die durch unterschiedliche Definitionen der Ausführung eines Befehls zustande kommen. Dies sind die hinterhältigsten Fehler, da sie nicht sofort sichtbar werden, sondern sich durch Nichtfunktionieren des Programms als ganzes oder durch unverständliche Fehlermeldungen äußern. Es werden auch hierzu ein paar Beispiele gegeben, die den Sachverhalt etwas klären:

ARRAYS

Bei Arrays besteht die häufigste Fehlerursache in der Definition des Index. Einmal ist als Index eine Zahl zugelassen, die größer als 0 sein muß, ein andermal kann die Zahl auch 0 sein. Dies führt im ersten Fall zu einer Fehlermeldung beim Starten des Programms oder erst bei einer bestimmten Eingabesituation. Zu beachten ist bei den Arrays noch, ob der Index in einer eckigen oder runden Klammer angegeben werden muß.

STRINGARRAYS

Manchmal findet man in einem Programm den Befehl DIM A$ (80). Hierbei ist zunächst noch nicht klar, ob es sich um ein Array von 80 Strings handelt oder um einen String mit 80 Zeichen. Auch das hängt von dem verwendeten BASIC ab. Bei BASIC-Versionen, bei denen es 80 Zeichen pro String bedeutet, sind meist auch die Befehle MID$ etc. nicht vorhanden, da durch den Index die Position eines Zeichens angegeben werden kann.

FOR TO NEXT

Hier kann ein besonders gefährlicher Fehler vorliegen. Ist beispielsweise der Wert bei den Schleifenparametern so, daß die Schleifenbedingung sofort erfüllt ist, z. B.

```
10 A = 10 · B = 5
20 FOR I=A TO B STEP 2
30 ...
40 NEXT I
```

dann gibt es zwei Möglichkeiten: 1. es wird die Schleife genau einmal durchlaufen, 2. die Schleife wird nicht durchlaufen, und das Programm wird nach dem NEXT-Befehl fortgesetzt. Dies kann zu einer sehr langwierigen Fehlersuche führen, und es ist wichtig, zu überprüfen, ob bei einem übernommenen Programm ein solcher Fall vorkommen kann (gewollt), und wie dann die Schleife ausgeführt werden soll. Zur Behebung eines entsprechenden Fehlers muß eine Abfrage der Schleifenparameter vor den FOR-Befehl eingebaut werden.

Mehrfachzuweisung

Auch hier können Inkompatibilitäten vorliegen. Z. B., es liegt die Zeile 10 A = B = 10 vor: Nun kann es passieren, daß beim eigenen BASIC Mehrfachzuweisungen nicht erlaubt sind, dabei aber logische Vergleiche in arithmetischen Ausdrücken. Dann erhalten die Variablen A und B nicht den Wert 10, sondern die Variable B bleibt unverändert, und die Variable A erhält 0, 1 oder -1. Denn die Zuweisung kann auch anders geschrieben werden: 10 A= (B =10). Hier wird deutlich, was geschieht. Zunächst wird B mit dem Wert 10 verglichen. Ist B =10, so erhält A den Wert „wahr", ist B<>10, so erhält A den Wert „falsch".

NOT, AND, OR

Hier kann es Schwierigkeiten durch die unterschiedliche Definition der Werte „*wahr*", und „*falsch*" geben. Einmal wird der Wert 1 für den Fall „*wahr*" ein andermal der Wert –1 verwendet. Schwierigkeiten können entstehen, wenn der Wert zur weiteren Verwendung arithmetisch aufgearbeitet wird.

Beispiel: 10 A =(C<D)*10. Ist C<D, so kann A einmal den Wert –10 erhalten, ein andermal den Wert 10.

Ferner gibt es noch ein anderes Problem. Erhält „*wahr*" den Wert 1 und bewirkt NOT eine bitweise Komplementbildung, so erhält die Variable A bei dem Befehl A =NOT(5 =5) den Wert 254. Bei der Abfrage IF A wird die IF-Anweisung ausgeführt. Würde „wahr" den Wert –1 erhalten, so träte dieses Problem nicht auf, denn NOT (–1) ergibt 0 also den Wert „*falsch*".

Rolf-Dieter Klein

Literatur

1 An Evaluation of three 8080 8K BASICs. Creative Computing Nov./Dec. 1977, Bd. 3, H. 6, S. 48 ff.
2 BASICally, BASIC. BYTE 1977, Bd. 2., H. 6, S. 96 ff.

Rolf-Dieter Klein

BASIC für 8080-Systeme

Der Neuling auf dem Mikrocomputergebiet kommt am schnellsten zu einem lauffähigen Programm, wenn er eine höhere Programmiersprache zur Verfügung hat. Mit dem beschriebenen Interpreter haben Besitzer von 8080- und Z-80-Systemen die Möglichkeit, in BASIC, einer der leichtesten Programmiersprachen überhaupt, zu arbeiten. Der Interpreter ist ein Übersetzungsprogramm, das aus den BASIC-Befehlen den Maschinencode des Mikroprozessors erzeugt. Er braucht etwa 3 KByte an Speicherplatz und kann zur Not noch von Hand eingetippt werden. Zusätzlich muß noch ungefähr 1 KByte RAM für das Benutzerprogramm zur Verfügung stehen. Weitere Voraussetzungen sind: alphanumerische Eingabe- und Ausgabemöglichkeit.

1 Arbeitsweise des BASIC-Interpreters

Grundsätzlich gibt es bei der Realisierung einer höheren Programmiersprache auf einem Mikrocomputer zwei verschiedene Möglichkeiten: Da wären zunächst einmal die sogenannten Compiler. Sie übersetzen ein Programm direkt in Maschinensprache und führen es dann aus. Das heißt, für einen BASIC-Befehl werden zunächst mehrere Befehle aus der Assemblersprache des betreffenden Prozessors erzeugt, und es entsteht aus dem ursprünglichen BASIC-Programm ein Programm in Assembler. Dieses wird dann in die entsprechenden Operationscodes übersetzt, und es entsteht ein für den Prozessor verständliches Programm, das in den Arbeitsspeicher geladen und gestartet werden kann. Nun erst läuft das Programm, und eventuelle Fehler werden erkennbar, wenn sie nicht schon vom Übersetzer erkannt wurden.

Ein Interpreter arbeitet etwas anders. Der vorliegende holt sich z. B. eine Zeile aus dem Programm, analysiert sie und führt die entsprechenden Befehle aus. Nach Ausführung der Zeile holt er die nächste und so fort. Ist eine Schleife vorhanden, das heißt, wird ein Programmteil öfters wiederholt, so muß der Interpreter diesen Teil auch genauso oft übersetzen. Hier liegt auch schon der Nachteil eines Interpreters gegenüber einem Compiler: Compiler-Programme sind in der Regel erheblich schneller. Es gibt noch eine Mischform, ebenfalls Interpreter, die das Quellprogramm in einen Zwischencode übersetzen und dann erst ausführen. Dabei werden Befehle wie PRINT einfach durch einen Code, z. B. 85, ersetzt. Der Vorteil dieser Interpreter liegt darin, daß Programme weniger Speicherplatz benötigen und daß die Ausführungszeit etwas besser liegt. Der beschriebene Interpreter arbeitet aber nicht so, sondern übersetzt jeden Befehl neu. Er besitzt dazu eine Tabelle, in der sämtliche vorhandenen Befehle stehen. Hinter jedem Befehl steht dann noch eine Adresse. Sie gibt an, an welcher Stelle im Programm sich das entsprechende Unterprogramm für die Abarbeitung des jeweiligen Befehls befindet.

2 Laden, Starten und Modifizieren

2.1 Laden des Interpreters

Um den Interpreter in das System eingeben zu können, sind zwei Möglichkeiten der Eingabe vorhanden. Zunächst ist eine sedezimale Liste (*Bild 1*) vorhanden. Der Interpreter startet auf der Adresse 1000. Die Codes können nun einfach entsprechend der Liste in den eigenen Computer eingegeben werden. Am Schluß sollte dabei jedes Byte mit dem aufgelisteten Byte verglichen werden, um Tippfehler, die das Programm zerstören könnten, zu vermeiden. Ferner sollte das Programm dann auf ein externes Speichermedium gebracht werden (z. B. Lochstreifen, Kassette o. ä.).

Es gibt aber noch eine zweite Möglichkeit. Dafür ist ein Listing im sogenannten „relocating"-Format beigefügt (*Bild 2*), das es gestattet, den Interpreter auf eine beliebige Stelle im Speicher (allerdings $<8000_{sed.}$) zu binden. Dies ist möglich, da in diesem besonderen Format die Information enthalten ist, an welchen Stellen Adressen im Programm vorhanden sind.

Um von diesem Format Gebrauch machen zu können, ist es aber nötig, einen sogenannten Lader zu schreiben, der diese Berechnung (addieren einer Konstanten auf alle Adressen) durchführt.

Dazu das Aufzeichnungsformat:

Zeichen 0...1: CR (Wagenrücklauf), LF (Zeilenvorschub), um Blöcke voneinander zu trennen.

Zeichen 2: Strichpunkt, kennzeichnet den Anfang eines relokalisierbaren Blockes.

Zeichen 3...4: Dieses Byte (durch zwei ASCII-Zeichen, ISO-7-bit-Code nach DIN 66003, 0...9, A...F dargestellt) gibt die Anzahl der Daten-Bytes an. Z. B. 19 an dieser Stelle bedeutet, es sind 25 Daten-Bytes in diesem Block.

Zeichen 5...8: In den zwei Bytes steckt die relative Anfangsadresse des Blocks. Sie wird beginnend mit dem höherwertigen Byte der Adresse angegeben.

Zeichen 9...10: Dieses Byte enthält die Relokalisierinformation. Ist dieses Byte 0, so wird die angegebene Anfangsadresse absolut verwendet. Ist dieses Byte 1,

so wird die Adresse durch den Relokalisierfaktor modifiziert.

Zeichen 11...12: Dieses Byte enthält die Relokalisierinformation für die nächsten 8 Daten-Bytes. Dabei entspricht jedes Bit dieses Bytes einem Byte der nächsten 8 Daten-Bytes. Bit 7 ist für das erste Byte zuständig und Bit 0 für das letzte. Ist das betreffende Bit 0, dann heißt dies, das Byte wird unverändert geladen. Ist ein Byte auf 1 gefolgt von einem auf 0, dann sind die beiden dazugehörigen Bytes eine relokalisierbare Adresse, und der Relokalisierfaktor muß aufaddiert werden. Ein Bit auf 1 gesetzt, von einem gefolgt, das ebenfalls auf 1 ist, stellt eine Information für den *Linking Loader* dar, der aber hier nicht benötigt wird.

Zeichen 13...28: Hierin sind die eigentlichen Daten-Bytes enthalten. Die Gesamtzahl der Daten-Bytes ist durch die Information am Anfang bestimmt. Nach jedem achten wird aber wieder die Relokalisierinformation eingefügt, dann folgen wieder Daten-Bytes.

Zeichen N...(N+1): Dieses Byte enthält die Blockprüfsumme, die dem Zweierkomplement der Summe über alle vorhergehenden Bytes in diesem Block ent-

```
1000  31 19 1B 3E FF C3 CA 16 E3 CD 28 10 BE C3 6B 10
1010  3E 0D C5 F5 3A 2E 1A B7 C3 FE 16 CD B5 13 E5 C3
1020  71 13 7C BA C0 7D BB C9 1A FE 20 C0 13 C3 28 10
1030  F1 CD 18 15 C3 31 15 CD 28 10 D6 40 D8 C2 5B 10
1040  13 CD 6C 14 29 DA A4 10 D5 EB CD B3 14 CD 22 10
1050  DA 61 15 2A 52 1B CD D6 14 D1 C9 FE 18 3F D8 13
1060  21 1B 1B 07 85 6F 3E 00 8C 67 C9 23 CA 76 16 C5
1070  4E 06 00 09 C1 1B 13 23 E3 C9 21 00 00 44 CD 28
1080  10 FE 30 D8 FE 3A D0 3E F0 A4 C2 A4 10 04 C3 44
1090  4D 29 29 09 29 1A 13 E6 0F 85 6F 3E 00 8C 67 C1
10A0  1A F2 81 10 D5 11 AB 10 C3 35 15 43 4F 57 3F 0D
10B0  0A 52 45 41 44 59 0D 0A 57 48 41 54 3F 0D 0A 53
10C0  4F 52 52 59 0D 0A 31 19 1B CD 10 10 11 B1 10 97
10D0  CD D4 15 21 DA 10 22 2F 1A 21 00 00 22 37 1A 22
10E0  31 1A 3E 3E CD 67 15 D5 CD 91 18 CD 7A 10 CD 28
10F0  10 7C B5 C1 CA 5E 18 1B 7C 12 1B 7D 12 C5 D5 79
1100  93 F5 CD AA 15 D5 C2 19 11 D5 CD C8 15 C1 2A 43
1110  1A CD 6B 16 60 69 22 43 1A C1 2A 43 1A F1 E5 FE
1120  03 CA C6 10 85 6F 3E 00 8C 67 CD 99 16 CD 22 10
1130  D2 60 15 22 43 1A D1 CD 76 16 D1 E1 CD 6B 16 C3
1140  E2 10 CD 2B 15 21 58 1B 22 43 1A CD 2B 15 C3 C6
1150  10 CD 2B 15 11 58 1B 21 09 00 CD B2 15 DA C6 10
1160  EB 22 2F 1A EB 13 13 CD 1C 1A 21 6F 17 C3 61 18
1170  CD 1B 10 D5 CD 2B 15 CD AA 15 C2 A5 10 F1 C3 60
1180  11 CD 7A 10 CD 2B 15 CD AA 15 DA C6 10 CD 36 16
1190  CD 1C 1A CD B2 15 C3 8A 11 0E 06 CD 08 10 3B 06
11A0  CD 10 10 C3 67 11 CD 08 10 0D 06 CD 10 10 C3 57
11B0  11 CD 08 10 23 07 CD 1B 10 40 C3 C3 11 CD E2 13
11C0  C3 D4 11 CD 08 10 2C 06 CD 15 13 C3 B1 11 CD 10
11D0  10 CD 30 10 CD 1B 10 C5 CD 12 16 C1 C3 C3 11 CD
11E0  A1 16 CD 1B 10 D5 CD AA 15 C2 A5 10 2A 2F 1A E5
11F0  2A 31 1A E5 21 00 00 22 37 1A 39 22 31 1A C3 60
1200  11 CD 2B 15 2A 31 1A 7C B5 CA 31 15 F9 E1 22 31
1210  1A E1 22 2F 1A D1 CD 85 16 CD 30 10 CD A1 16 CD
1220  FF 14 2B 22 37 1A 21 2E 18 C3 61 18 CD 1B 10 22
1230  3B 1A 21 36 18 C3 61 18 CD 1B 10 C3 41 12 21 01
1240  00 22 39 1A 2A 2F 1A 22 3D 1A EB 22 3F 1A 01 0A
1250  00 2A 37 1A EB 60 68 39 3E 09 7E 23 B6 CA 7A 12
1260  7E 2B BA C2 59 12 7E BB C2 59 12 EB 21 00 00 39
1270  44 4D 21 0A 00 19 CD 76 16 F9 2A 3F 1A EB CD 30
1280  10 CD 37 10 DA 31 15 22 33 1A D5 EB 2A 37 1A 7C
1290  B5 CA 32 15 CD 22 10 CA A4 12 D1 CD 83 16 2A 33
12A0  1A C3 8A 12 5E 23 56 2A 39 1A E5 7C AA 7A 19 FA
12B0  B6 12 AC FA DA 12 EB 2A 37 1A 73 23 72 2A 3B 1A
12C0  F1 B7 F2 C6 12 EB CD F5 14 D1 DA DC 12 2A 3D 1A
12D0  22 2F 1A 2A 3F 1A EB CD 36 10 E1 D1 CD 85 16 CD
12E0  30 10 21 00 00 C3 EB 12 CD 1B 10 7C B5 C2 67 11
12F0  CD CA 15 D2 60 11 C3 C6 10 2A 35 1A F9 E1 22 2F
1300  1A D1 D1 D5 CD E2 15 C3 13 13 CD 37 10 DA 31 13
1310  C3 25 13 D5 CD 37 10 DA 31 15 1A 4F 97 12 D1 CD
1320  D4 15 79 1B 12 D5 EB 2A 2F 1A E5 21 03 13 22 2F
1330  1A 21 00 00 39 22 35 1A D5 3E 3A CD 67 15 CD 91
1340  18 CD 1B 10 CD 1C 1A D1 EB 73 23 72 E1 22 2F 1A
1350  D1 F1 CD 08 10 2C 03 C3 03 13 CD 30 10 1A FE 0D
1360  CA 6E 13 CD FF 14 CD 08 10 2C 03 C3 63 13 CD 30
1370  10 21 40 18 C3 61 18 CD A0 13 D8 6F C9 CD A0 13
1380  C8 6F C9 CD A0 13 C8 D8 6F C9 CD A0 13 6F C8 D8
1390  6C C9 CD A0 13 C0 6F C9 CD A0 13 D0 6F C9 E1 C9
13A0  79 E1 C1 E5 C5 4F CD B5 13 EB E3 CD 75 14 D1 21
13B0  00 00 3E 01 C9 CD 08 10 2D 06 21 00 00 C3 E7 13
13C0  CD 08 10 2B 00 CD F1 13 CD 08 10 2B 15 E5 CD F1
13D0  13 EB E3 7C AA 7A 19 D1 FA C8 13 AC F2 C8 13 C3
13E0  A4 10 CD 08 10 2D 92 E5 CD F1 13 CD E0 14 C3 D1
13F0  13 CD 55 14 CD 08 10 2A 2D E5 CD 55 14 06 00 CD
1400  DD 14 E3 CD DD 14 EB E3 7C B7 CA 13 14 7A B2 EB
1410  C2 A5 10 7D 21 00 00 B7 CA 47 14 19 DA A5 10 3D
1420  C2 1B 14 C3 47 14 CD 08 10 2F 4E E5 CD 55 14 06
1430  00 CD DD 14 E3 CD DD 14 EB E3 EB 7A B3 CA A5 10
1440  C5 CD C0 14 60 69 C1 D1 7C B7 FA A4 10 78 B7 FC
1450  E0 14 C3 F4 13 21 ED 17 C3 61 18 CD 37 10 DA 66
1460  14 7E 23 66 6F C9 CD 7A 10 78 B7 C0 CD 08 10 28
1470  09 CD 1B 10 CD 08 10 29 01 C9 C3 31 15 CD 6C 14
1480  7C B7 FA A4 10 B5 CA A4 10 D5 E5 2A 41 1A 11 2D
1490  1A CD 22 10 DA 9A 14 21 00 10 5E 23 56 22 41 1A
14A0  E1 EB C5 CD C0 14 C1 D1 23 C9 CD 6C 14 1B CD DD
14B0  14 13 C9 2A 43 1A D5 EB 2A 52 1B CD D6 14 D1 C9
14C0  E5 6C 26 00 CD CB 14 41 7D E1 67 0E FF 0C CD D6
14D0  14 D2 CD 14 19 C9 7D 93 6F 7C 9A 67 C9 7C B7 F0
14E0  7C B5 C8 7C F5 2F 67 7D 2F 6F 23 F1 AC F2 A4 10
14F0  78 EE 80 47 C9 7C AA F2 FB 14 EB CD 22 10 C9 CD
1500  37 10 DA 31 15 E5 CD 08 10 3D 0A CD 1B 10 44 4D
1510  E1 71 23 70 C9 C3 31 15 CD 08 10 3B 04 F1 C3 67
1520  11 CD 08 10 0D 04 F1 C3 57 11 C9 CD 28 10 FE 0D
1530  C8 D5 11 B8 10 97 CD D4 15 D1 1A F5 97 12 2A 2F
1540  1A E5 7E 23 B6 D1 CA C6 10 7E B7 FA F9 12 CD 56
1550  16 1B F1 12 3E 3F CD 12 10 97 CD D4 15 C3 C6 10
1560  D5 11 BF 10 C3 35 15 CD 12 10 CD 91 18 CD 1D 17
1570  FE 01 CA 92 15 CD 12 10 FE 0A CA 6D 15 B7 CA 6D
1580  15 FE 1B CA A2 15 12 13 FE 0D C8 7B CD A1 18 C2
1590  6D 15 7B CD A8 18 CA A2 15 1B 3E 08 CD 12 10 C3
15A0  6D 15 CD 10 10 3E 0B C3 67 15 7C B7 FA A4 10 11
15B0  58 1B E5 2A 43 1A 2B CD 22 10 E1 D8 1A 95 47 13
15C0  1A 9C DA C9 15 1B B0 C9 13 13 1A FE 0D C2 C9 15
15D0  13 C3 B2 15 47 1A 13 B8 C8 CD 12 10 FE 0D C2 D5
15E0  15 C9 CD 08 10 22 0F 3E 22 CD D4 15 FE 0D E1 CA
15F0  57 11 23 23 23 E9 CD 08 10 27 05 3E 27 C3 E9 15
1600  CD 08 10 5F 0C 3E 8D CD 12 10 CD 12 10 E1 C3 F2
1610  15 C9 06 00 CD DD 14 F2 1D 16 06 2D 0D D5 11 0A
1620  00 D5 0D C5 CD C0 14 78 B1 CA 34 16 E3 2D E5 60
1630  69 C3 24 16 C1 0D 79 B7 FA 43 16 3E 20 CD 12 10
1640  C3 35 16 78 B7 C4 12 10 5D 7B FE 0A D1 C8 C6 30
1650  CD 12 10 C3 49 16 1A 6F 13 1A 67 13 0E 04 CD 12
1660  16 3E 20 CD 12 10 97 CD D4 15 C9 CD 22 10 C8 1A
1670  02 13 03 C3 6B 16 78 92 C2 7E 16 79 93 C8 1B 2B
1680  1A 77 C3 76 16 C1 E1 22 37 1A 7C B5 CA 9F 16 E1
1690  22 39 1A E1 22 3B 1A E1 22 3D 1A E1 22 3F 1A C5
16A0  C9 21 4F 1A CD E0 14 C1 39 D2 60 15 2A 37 1A 7C
16B0  B5 CA C7 16 2A 3F 1A E5 2A 3D 1A E5 2A 3B 1A E5
16C0  2A 39 1A E5 2A 37 1A E5 C5 C9 32 2E 1A 16 03 CD
16D0  10 10 15 C2 CF 16 97 11 37 17 CD D4 15 21 00 10
16E0  22 41 1A 21 58 1B 22 43 1A 21 B2 1D 22 52 1B 21
16F0  B4 1D 22 54 1B 21 F4 1D 22 56 1B C3 C6 10 C2 04
1700  17 F1 C1 C9 F1 F5 4F 79 FE 0D CA 13 17 CD 09 F0
1710  F1 C1 C9 0E 0D CD 09 F0 0E 0A C3 0D 17 CD 03 F0
1720  E6 7F FE 02 C2 31 17 3A 2E 1A 2F 32 2E 1A C3 1D
1730  17 FE 03 C0 C3 C6 10 52 44 4B 20 50 52 4F 4D 50
1740  54 20 42 41 53 49 43 20 56 33 2E 31 20 33 4B 0D
1750  0A 4C 49 53 54 00 81 11 52 55 4E 00 51 11 4E 45
1760  57 00 42 11 42 59 45 00 DF 18 45 4E 44 00 AF 18
1770  4E 45 58 54 00 81 12 4C 45 54 00 63 13 49 46 00
1780  E8 12 47 4F 54 4F 00 70 11 47 4F 53 55 42 00 DF
1790  11 52 45 54 55 52 4E 00 01 12 52 45 4D 00 E2 12
17A0  46 4F 52 00 1C 12 49 4E 50 55 54 00 03 13 50 52
17B0  49 4E 54 00 99 11 53 54 4F 50 00 4B 11 43 41 4C
17C0  4C 00 E3 18 4F 55 54 43 48 41 52 00 C1 19 4F 55
17D0  54 00 F0 18 4F 24 00 41 19 49 24 00 4E 19 50 4F
17E0  4B 45 00 87 19 54 41 42 00 16 19 00 5D 13 52 4E
17F0  44 00 7D 14 41 42 53 00 AA 14 53 49 5A 45 00 B3
1800  14 50 45 45 4B 00 80 19 49 4E 43 48 41 52 00 CB
1810  19 48 45 58 00 D2 19 49 4E 00 27 19 27 00 A9 19
1820  54 4F 50 00 B7 19 4C 45 4E 00 BC 19 00 5B 14 54
1830  4F 00 2C 12 00 31 15 53 54 45 50 00 38 12 00 3E
1840  12 3E 3D 00 77 13 23 00 7D 13 3E 00 83 13 3D 00
1850  92 13 3C 3D 00 8A 13 3C 00 98 13 00 9E 13 21 50
1860  17 CD 28 10 D5 1A 13 FE 2E CA 83 18 23 BE CA 65
1870  18 3E 00 1B BE CA 8A 18 23 BE C2 78 18 23 23 D1
1880  C3 61 18 3E 00 23 BE C2 85 18 23 7E 23 66 6F F1
1890  E9 E5 2A 54 1B 54 5D E1 C9 E5 2A 52 1B 54 5D E1
18A0  C9 E5 2A 56 1B BD E1 C9 E5 2A 54 1B BD E1 C9 CD
18B0  1B 10 EB 21 B2 1D EB CD 22 10 DA 61 15 7D B7 FA
18C0  61 15 7E 2F 77 46 B8 C2 61 15 22 56 1B 7D D6 84
18D0  6F 7C DE 00 67 22 54 1B 2B 22 52 1B C3 C6 10 FF
18E0  C3 C6 10 CD 1B 10 D5 01 EC 18 C5 E9 D1 CD 30 10
18F0  CD 6C 14 E5 CD 08 10 3D 1A CD 1B 10 45 3E D3 32
1900  2A 1A E1 7D 32 2B 1A 3E C9 32 2C 1A 78 CD 2A 1A
1910  CD 30 10 C3 31 15 CD 6C 14 7C B5 CC 30 10 2B 3E
1920  20 CD 12 10 C3 19 19 CD 6C 14 E5 3E DB 32 2A 1A
1930  E1 7D 32 2B 1A 3E C9 32 2C 1A CD 2A 1A 26 00 6F
1940  C9 CD 1B 10 D5 EB AF CD D4 15 D1 CD 30 10 CD 1B
1950  10 D5 EB 2A 43 1A EB CD 22 10 DA 61 15 CD 91 18
1960  CD 6D 15 44 4D EB 2B CD 91 18 D5 CD 6B 16 AF 02
1970  D1 23 CD D6 14 EB 21 28 1A 73 23 72 D1 CD 30 10
1980  CD 6C 14 6E 26 00 C9 CD 1B 10 D5 EB 2A 43 1A EB
1990  CD 22 10 DA 61 15 D1 E5 CD 08 10 2C 09 CD 1B 10
19A0  7D E1 77 CD 30 10 C3 31 15 1A 13 6F 26 00 CD 08
19B0  10 27 01 C9 C3 31 15 2A 43 1A 23 C9 2A 28 1A 2B
19C0  C9 CD 1B 10 7D CD 12 10 CD 30 10 CD 1D 17 26 00
19D0  6F C9 C5 21 00 00 CD 08 10 28 1D 1A FE 0D CA 31
19E0  15 CD FD 19 29 29 29 29 06 00 4F 09 13 CD 08 10
19F0  29 03 C3 FB 19 C3 DB 19 C3 31 15 C1 C9 FE 30 FA
1A00  31 15 FE 39 FA 14 1A CA 14 1A FE 41 FA 31 15 FE
1A10  47 F2 31 15 D6 30 FE 0A F8 D6 07 C9 C9 00 00 00
1A20  00 00 06 00 00 00 00 C9 00 00 00 00 00 00 FF DA
1A30  10 00 00 2D 1B 00 00 00 00 01 00 0A 00 58 1B 67
1A40  1B 0A 10 82 1B 00 00 00 00 00 00 00 00 00 00 00
```

Bild 1. Programmliste

spricht, so daß nun die Summe aller Bytes 0 ergeben muß. Damit wird eine Blocksicherung erreicht.

Vom Benutzer muß ein Lader geschrieben werden, der speziell auf sein System abgestimmt ist und alle diese Informationen auswertet. Der Vorteil des relokalisierbaren Programms liegt neben der frei wählbaren Anfangsadresse bei der Überprüfbarkeit auf Eingabefehler durch die Prüfsumme.

2.2 *Starten und Modifizieren des Interpreters*

Um den Interpreter starten zu können, ist es nötig, zunächst die Ein/Ausgabe-Vektoren zu ändern. Standardmäßig sind zwei Vektoren vorhanden, der eine führt auf die Adresse F003 und bedeutet einen Sprung auf die Eingabe eines Zeichens vom Benutzer. Das Zeichen muß dabei über das Register A an den Interpreter weitergegeben werden. Die anderen Register dürfen nicht verändert werden. Ferner gibt es noch einen Sprung auf die Adresse F009, der die Ausgabe eines Zeichens bewirken soll. Dabei ist das auszugebende Zeichen im Register C vorhanden. Am Schluß dieser Routine muß das Zeichen im Register A stehen, und C soll unverändert bleiben, wie auch die restlichen Register.

Hat der Benutzer an diesen beiden Adressen keinen Speicher, um entsprechende Sprungbefehle auf seine eigenen E/A-Routinen durchzuführen, so muß er im BASIC-Interpreter die Vektoren ändern.

Dazu die Adressen: Auf Adresse 71DH (H bedeutet HEX also sedezimal) steht die Befehlsfolge CD 03 F0, sie ist in die entsprechende Folge zum Aufruf des eigenen Eingabe-Unterprogramms umzuwandeln. Auf Adresse 70DH und 715H steht die Befehlsfolge CD 09 F0, hier ist der entsprechende Aufruf für die Ausgaberoutine einzutragen.

Nun gibt es noch die Möglichkeit, das Anwenderprogramm durch die Betätigung der Tastenfolge CTRL C zu stoppen, genauso wie den Ausdruck langer Programme. Dazu hat der Benutzer eine Routine zu schreiben, die seinen Status-Port der Eingabetastatur abfragt, ob ein Zeichen vorhanden ist und wenn, ob es sich um CTRL C (Code 3) handelt. Ist das der Fall, muß

```
;1800000142311900B3EFFC3CA0622E3CD2800BEC36B00003E0DC5F53A5B
;18001501892E0AB7C3FE06CDB51003E5C371037CBAC0007DBBC91AFED3
;19002A010820C013C32800F1CD921805C33105CD280008D640D8C25B0068
;190040012213CD6C0429DAA40012D5EBCD8304CD220048DA61052A520B39
;1800560140CDD604D1C9FE1B3F10D813211B0B07856F003E008C67C981
;19006B012023CA7600C54E060009C11B1323E3C9210800044CD2800B6
;1800810100FE30D8FE3AD03EF020A4C2A40004C5444D00292909291A08
;1800960100013E60F854F3E008C0867C11AF26100D51170AB00C33505B0
;1800AB0100484F573F0D0A52450041445 90D0A57484100543F0D0A538F
;1900C001014F5252590D0A3119240BCD100011B1009748CDD40521DA002A
;1800D60141222F0A2100002237210A22310A3E3ECD671005D5CD910873
;1800EB0148CD7A00CD28007CB520C1CA5E08137C121B007D12C5D579D0
;180100011193F5CDAA05D5C2191101D5CDC805C12A43200ACD6B0660AA
;18011501226922430AC12A430A04F1E5FE03CAC60085006F3E008C670F
;18012A0149CD9900CD2200D260220522430AD1CD768610D1E1CD6B0634
;18013F0149C3F200CD2B052158240B22430ACD2B05C390C600CD2B0592
;190154014111580B210000CDB22205DAC600EB222F0A08EB1313CD1C0A23
;18016A0149216F07C36108CD1B1200D5CD2B05CDAA0540C2A500F1C3CD
;18017F01926001CD7A00CD2B0549CDAA05DAC600CD562806CD1C0ACDC2
;1801940190B205C38A010E06CD849893B06CD1390C3906701CD0800A2
;1801A901120D06CD1000C3570142CD08002307CD1D00284DC3C301CD36
;1901BE0192E205C3D401CD0800122C06CD1805C3B10148CD1000CD30007C
;1801D40144CD1B00C5CD1206C149C5C301CDA106CD1B1000D5CDAA05EE
;1801E90148C2A5002A2F0AE52A81310AE52100002237100A3922310A11
;1801FE0149C36001CD2B052A31080A7CB5CA3105F9E14022310AE12266
;1902130189 2F0AD1CD8506CD302400CDA106CDFF042B4822370A212E0854
;1802290149C36108CD1B00223B240A213608C36108CD901B00C34102CB
;19023E01092191002239 0A2A2F220A223D0AEB223F0A08010A002A370A54
;1802540100EB6068393E097E2320B6CA7A027E2BBAC20039027EBBC2A6
;18026901805902EB2100003944024D210A0017CD760620F92A3F0AEBC5
;19027E0149CD3000CD3700DA31210522330AD5EB2A37000A7CB5CA320527
;1902940148CD2200CAA402D1CD9285062A330AC38A02085E23562A390AEC
;1902AA0102E57CAA7A19FAB60222ACFADA02EB2A370A087323722A3B0A6F
;1902C000111F1B7F2C602EBCDF51204D1DADC022A3D0A40222F0A2A3F0ADE
;1802D60121EBCD3000E1D1CD852006CD3000210000C390EB02CD1B0096
;1902EB01127CB5C26701CDCA0549D26001C3C6002A3080AF9E1222F0A45
;1903010109D1D1D5CDF205C3132403CD3700DA5103C3C82503D5CD370033
;18031701400A31051A4F9712D140CDD405791B12D5EB402A2F0AE521A5
;18032C01900303222F0A2100002039223 50AD53E3ACD906705CD910670
;1803410148CD1B00CD1C0AD1EB04732372E1222F0AD120F1CD08002C99
;180356012403C30303CD360001A12FE0DCA6E03CDF10440CD08002C631B
;18036B0149C36303CD300021492408C36108CDA003D8106FC9CDA00351
;1803800108C86FC9CDA003C8D8106FC9CDA0036FC8D8106CC9CDA003D0
;1803950108C06FC9CDA003D06F00C9E1C979E1C1E5C5204FCDB503EB89
;1903AA0120E3CDF504D1210000683E01C9CD08002D0608210000C3E70390
;1803C00142CD08002800CDF10341CD08002B15E5CDF100032BE37CAA31
;1903D501687A19D1FAC803ACF292C803C3A160CD0800082D92E5CDF10339
;1803EB0149CDE004C3D103CD552004CD08002A2DE5CD8035040600CD78
;18040001883D04E3CDD041BE3107CB7CA13047AB2EB40C2A5007D219B
;18041501080000B7CA470419DAB9A5003DC21B04C5472004CD3002F89
;19042A01104EE5CD55040600CD08DD04E3CDDD04EBE308EB7AB3CAA50025
;1804400120C5CDC0044A069C1D1107CB7FAA40073B7FC99E004C3F40378
;180455014921ED07C36108CD372000DA66047E23666F20C9CD7A00787E
;19046A0110B7C0CD08002807CD701B00CD08002701C748C33105CD6C042D
;18048001117CB7FAA40085CAA40700D5E52A310A112D200ACD2200DAF5
;18049501909A042109905E23564122410AE1EBC5CDC86004C1D123C9DA
;1804AA0144CD6C041BCDD0041321C92A430AD5EB2A22200BCDD604D19C
;1804BF0102C9E56C2600CDCB0400417DE167EEFF0CCD70D604D2CD044D
;1804D4010019C97D936F7C9A6700C97CB7F07CB5C87C06F52F677D2F99
;1804E901046F23F1ACF2A400730 1EE3647C97CAAF2F51004EBCD220039
;1804FE0124C9CD3700DA3105E542CD08003D0ACD1B000444DE17123E3
;1805130112 70C9C33105CD08000093B04F1C36701CD08000000D04F1C3B8
;18052801885701C9CD2800FE0D11C8D511B80097CDD40005D11AF597E6
;18053D0120122A2F0AE57E23B621D1CAC6007EB7FAF92002CD56061BC4
;1805520104F1123E3FCD12009748CDD405C3C600D51190BF00C33505ED
;1905670149CD1200CD9108CD1D0907FE01CA9205CD120800FE0ACA6D0567
;18057D0121B7CA6D05FE1BCAA2000512 13FE0DC87BCD90A108C26D051A
;18059201247BCDA809CAA2051B123E09CD1200C36D0540CD10003E0BD6
;1805A70142C367057CB7FAA4004411580BE52A430A2B40CD2200E1D8D2
;1805BC01011A9547131A9CDAC900051BB0C913131AFE200DC2C905131C
;1805D10140C3B205471A13B8C842CD1200FE0DC2D50520C9CD080022BB
;1805E601080F3E22CDD405FE0D20E1CA5701232323E940CD080027051E
;1905FB01123E27C3E905CD080045F0C3E8DCD1200CD881200E1C3F205CE
;1906110109C90600CDDD04F21D0006062D0DD05110A0008D50DC5CDC004C4
;180627011078B1CA3406E32DE5106069C32406C10D7920B7FA43063E23
;18063C012420CD1200C335067820B7C412005D7BFE0A00D1C8C630CD23
;1806510190120 0C349061A6F13021A67130E04CD1206103E20CD120066
;180666012297CDD405C9CD220002C81A021303C36B06107892C27E06D4
;18067B01007993C81B2B1A77C3847606C1E122370A7C20B5CA9F06E152
;180690014422390AE1223B0AE144223D0AE1223F0AC520C9214F0ACD91
;1906A50184E004C139D260052A84370A7CB5CAC7062A863F0AE52A3D0A9A
;1806BB0122E52A3B0AE52A390A20E52A370AE5C5C932802E0A1603CDAB
;1906D0018810001 5C2CF069711923707CDD4052100004822410A21580B54
;1806E6014922430A21B20D2252240B21B40D22540B2190F40D22560B28
;1806FB0148C3C600C20407F1C100C9F1F54F79FE0DCA801307CD09F0EA
;18071000100F1C1C90FE0DCD09F0100E0AC30D07CD03F000E67FFE02C28E
;180725019131073A2E0A2F322E200AC31D07FE03C0C380C60052444B35
;18073A010020505 24F4D05420004241534943205633002E3120334B7C
;18074F01010D0A4C4953540081040152554E0051014E10455700420133
;18076401084259450 0DF08454E204400AF084E455854400081024C456C
;18077901215400650349 4600E80202474F544F00700100474F535542E7
;18078E014000DF0152455455 52204E09010252454D0080E202464F5200
;1807A30140001C02494F50555440000303505249 4E5440069901 5354FB
;1907B801104F50004B0143414C204C00E3084F5554430848415200C1091D
;1807CE01084F555400F0084F244200410949 24004E0900504F4B450028
;1807E30182870954414200160941005D03524E44007D00044142530019
;1807F80181AA0453495A4500B302045045454B0000900474E434841B4
;18080D01205200CB0948455800840289494E002709274000A909544FC5
;18082201205000B7094C454E0090BC09005B04544F00902C0200310563
;18083701045354455000380200843E023E3D0077032340007D033E0054
;18084C01883033D0092033C3D44008A033C00980300909E03215007E9
;1808610140CD2800D51A13FE2E42CA830823BECA6508003E001BBECA8B
;18087601848A0823BEC27800231023D1C361083E002320BEC28508232C
;18088B01007E23666FF1E9E52A80540B545DE1C9E52A80520B545DE13D
;1908A00110C9E52A560BBDE1C920E52A540BBDE1C9CD881B00EB21B20D5E
;1808B60124EBCD2200DA61057D20B7FA61057E2F77462088C2610522AB
;1808CB0180560B7DD6846F7CDE11006722540B262252200BC3C600FF48
;1808E00148C3C600CD1B00D50182EC08C5E9D1CD300040CD6C04E5CD4F
;1808F5018408003D2ACD1B0045103ED3322A0AE17D32802B0A3EC932D5
;18090A01892C0A78CD2A0ACD302400C33105CD6C047C20B5CC30002BCD
;19091F01123E20CD1200C3190941CD6C04E53EDB322A080AE17D322B0ADB
;18093501123EC9322C0ACD2A0A0426006FC9CD1B00D510EBAFCDD405B8
;19094A0124D1CD3000CD1B00D522EB2A430AEBCD220048DA6105CD910898
;1809600140CD6D05444DEB2BCD889108D5CD6B06AF0210D123CDD604FB
;1809750120EB21280A732372D148CD3600CD6C046E261000C9CD1B005B
;19098A0111D5EB2A430AEBCD222100DA6105D1E5CD080802C09CD1B0020
;1809A001097DE177CD3000C33101051A136F2600CD0800002701C9C31E
;1909B5019031052A430A23C92A84280A2BC9CD1B007D48CD1200CD3000A2
;1909CB0140CD1D0726006FC9C508210000CD0800281D081AFE0DCA31054E
;1809E10140CDFD09292929290604004F0913CD0800292003C3FB09C326
;1809F60190DB09C33105C1C9FE2130FA3105FE39FA14200ACA140AFE1D
;150A0B012141FA3105FE47F2310005D630FE0AF8D60700C9C961
;020A270100C903
;0000000000
```

Bild 2. Programmliste für verschiebbares (relocating) Format

das Programm auf die Adresse 00C6H springen (RESTART-Adresse des Interpreters), anderenfalls muß es mit einem Return-Befehl zurückkehren. Diese Routine wird von zwei Stellen des BASIC-Interpreters aus angesprungen. Der Aufruf muß an die betreffenden Stellen eingetragen werden.

Dazu die Adresse: Auf A1CH steht der Befehl C9 (Return), dieser muß durch einen JMP-Befehl auf die eigene Routine ersetzt werden. Falls der Platz ausreicht, kann die Routine auch an diese Stelle geschrieben werden. Es stehen dazu 10 Byte zur Verfügung. Die Adresse A1CH wird von dem Teil, der die Programme ausführt, und von dem Teil, der Programme auflistet, bei Ausführung jedesmal mit dem Beginn einer neuen Zeile angesprungen.

Im beschriebenen BASIC gibt es einen Befehl BYE, der das System verläßt und dem Monitor die Steuerung übergibt. Dies geschieht durch die Ausführung des Maschinenbefehls RST 7. Der Benutzer kann einen direkten Sprung zu seinem Monitor einbauen, indem er an die Stelle 8DFH anstelle des RST 7-Befehls einen Sprung zu seinem Monitor schreibt. Das BASIC-Programm kann dann vom Monitor aus gestartet werden, indem ein Sprung auf den Speicherplatz C6 ausgeführt wird.

Alle angegebenen Adressen sind um den Relokalisierfaktor zu erhöhen, den der Anwender beim Laden des Interpreters verwendet hat.

3 BASIC-Befehle

3.1 Steuerbefehle

Diese Befehlsgruppe dient der Steuerung des Interpreters. Es kann damit der Ablauf eines Programms bestimmt werden (starten, löschen, ausdrucken).

LIST

Mit Hilfe dieses Befehls kann ein zuvor eingegebenes Programm ausgedruckt werden. Dabei wird, wenn der Befehl LIST CR (CR steht für *carriage return*, also Wagenrücklauf) eingegeben wird, das gesamte Programm ausgedruckt. Es kann auch eine zusätzliche Zeilennummer angegeben werden, dann wird das Programm beginnend mit dieser Zeilennummer ausgegeben. Soll der Ausdruck eines Programms abgebrochen werden, weil auf dem Datensichtgerät nicht das ganze Programm dargestellt werden kann, so kann dies durch Eingeben des Zeichens CTRL C (Code 3) geschehen. LIST 200 zum Beispiel gibt das Programm beginnend mit der Zeilennummer 200 aus. Falls die Zeilennummer 200 nicht vorhanden ist, wird nach der nächstgrößeren gesucht und von da an gelistet.

RUN

Nach Eingabe von RUN CR wird das Programm beginnend bei der niedrigsten Zeilennummer gestartet.

NEW

Damit kann ein Programm gelöscht werden. Anschließend ist es möglich, ein neues Programm einzugeben.

BYE

Bei der Ausführung dieses Befehls kehrt der Prozessor zum Monitorprogramm zurück und verläßt das BASIC-System.

END

In der ursprünglichen Bedeutung steht dieser Befehl am Ende eines Programms. Dies ist bei dem vorliegenden Interpreter nicht notwendig, somit konnte dieser Befehl mit einer anderen Bedeutung versehen werden. Von Haus aus wird nach Laden des BASIC-Interpreters ein minimaler Speicherbedarf von 700 Byte für den Anwender definiert. Bei der Überschreitung dieses Arbeitsraumes durch Eingabe langer BASIC-Programme wird eine Fehlermeldung vom Interpreter (SORRY) ausgegeben, die darauf hinweisen soll, daß kein Platz mehr für das Programm vorhanden ist. Nun kann es aber sein, daß ein größerer Speicher vorhanden ist. In diesem Fall kann der Benutzer mit Hilfe des Befehls END den zunächst auf 700 Byte vorbesetzten Platz dynamisch erhöhen. Dazu erhält der Befehl END einen zusätzlichen Parameter, der die absolute Adresse der gewünschten neuen höchsten Adresse darstellt.

Zum Beispiel bedeutet die Anweisung END 8000, daß dem BASIC ab sofort Speicherplatz bis zur Adresse 8000 (dezimal) zur Verfügung steht.

Dabei muß berücksichtigt werden, daß der Interpreter darüber hinaus einen Platz für den Textpuffer benötigt, der über dem angegebenen Platz vorhanden sein muß. Er kann mit etwa 140 Byte veranschlagt werden.

3.2 Programmierbare Befehle

Alle nun folgenden Befehle können im Gegensatz zu den Steuerbefehlen programmiert werden. Die meisten können auch im sogenannten „Direct Mode" verwendet werden, das heißt, einfach durch Eingabe ohne vorangestellte Zeilennummer. Sie werden dann unmittelbar nach Eingabe von CR ausgeführt.

LET

LET weist einer Variablen einen Wert zu.

Beispiel:
```
10 LET A=10
20 LET B=2*(3-9)*6/2
30 LET A=C
```

Dabei kann der Befehl LET auch weggelassen werden. Er dient eigentlich nur der besseren Lesbarkeit.

FOR TO NEXT

Damit ist es möglich, Schleifen aufzubauen, das heißt, eine bestimmte Befehlsfolge n-mal zu durchlaufen.

Beispiel:
```
10 FOR A=1 TO 10 STEP 2
20 ...
30 ...
40 NEXT A
```

Der Bereich zwischen 10 und 40 wird dabei 5mal durchlaufen. Innerhalb der Schleife kann der aktuelle Wert von A verwendet werden, er sollte jedoch nicht verändert werden. Die Angabe STEP legt die Schritt-

Programmbeispiele

```
10 PRINT 'GEBEN SIE EINEN STRING EIN'
20 I$ TOP
30 FOR I=1 TO LEN
40 IF PEEK(TOP+I)=' ' POKE TOP+I,'-'
50 NEXT I
55 O$ TOP

>RUN
GEBEN SIE EINEN STRING EIN
DIES IST EIN TESTSTRING
DIES-IST-EIN-TESTSTRING
```

```
10 FOR I=1 TO 10
20 FOR J=1 TO 10
30 @(I)=@(I)+RND(100)
40 NEXT J
50 NEXT I
60 FOR I=1 TO 10
70 PRINT @(I),
80 NEXT I

>RUN
  479   612   269   685   379   496   514   477   338   422
READY
>RUN
  872  1048   917  1143   983   952  1057   834   899   684
READY
>RUN
 1468  1570  1358  1711  1500  1472  1664  1350  1332  1147
READY
```

```
>PRINT A,B,A<B,A>B,A=B,A<==
   32    13     0     1     0     0     0

READY
>PRINT A,B,A<B,A>B,A=B,A#B,A<=B,A>=B
   32    13     0     1     0     1     0     1

READY
>PRINT (1=1)*(A<B)*100+A+A*('1'='1')
   64
>PRINT SIZE
  602

READY
>END HEX(3F00)

READY
>PRINT SIZE
 8995
```

```
>10 REM AUSDRUCKEN VON ZUFALLSZAHLEN ZWISCHEN 0 UND 9
>20 FOR I=1 TO 5
>30 PRINT RND(10)-1
>40 NEXT I

>RUN
    9
    7
    9
    2
    5
```

```
>10 INPUT 'GEBEN SIE ZAHL EIN'A,B
>20 PRINT A*A,B*B,A+B
>30 PRINT 'MIT ANDEREM FORMAT'
>40 PRINT #10,A+A,B*B,A+B

>RUN
   GEBEN SIE ZAHL EIN:12+1
   B:23-5
      169    324     31
   MIT ANDEREM FORMAT
           26          324          31
```

```
10 REM ANWENDUNG VON INCHAR
20 A=INCHAR
30 IF A=' ' STOP
40 IF A='0' B='*'
50 IF A='1' B=' '
60 IF A='2' B=HEX(D)
70 OUTCHAR(B)
80 GOTO 20

>RUN
****   * *    * *
 *  ** **
```

```
 10 REM GAUSSCHE VERTEILUNG
 20 FOR I=1 TO 300
 30 FOR J=1 TO 10
 40 @(I)=@(I)+RND(60)
 50 NEXT J
 60 NEXT I
 70 FOR I=1 TO 300
 80 @(I)=@(I)/10
 90 NEXT I
100 A=0
105 I=1
110 IF @(I)<@(I+1) H=@(I+1);@(I+1)=@(I);@(I)=H;A=1
120 I=I+1
130 IF I<300 GOTO 110
140 IF A=1 GOTO 100
150 FOR I=1 TO 300
160 IF @(I)=@(I+1) PRINT " ",
170 IF @(I)>@(I+1) PRINT "*"
180 NEXT I

>RUN
  *
   *
    *
         *
        *
        *
          *
           *
           *
              *
              *
              *
                      *
                    *
                     *
                   *
                *
                  *
                *
                    *
                        *
               *
      *
     *
     *
   *
   *
  *
  *
  *

  READY

  REM ZEITBEDARF ETWA 45 MIN
```

```
>REM DRUCKEN IN EINER ZEILE

READY
>10 FOR I=1 TO 5
>20 PRINT RND(10)-1,
>30 NEXT I

>RUN
     7     4     4     7     5
READY
```

weite fest. Sie kann auch negativ sein, es müssen dann allerdings auch die Variablen nach FOR und TO entsprechend gewählt werden.

Beispiel:
```
10 FOR A=10 TO 1 STEP -2
20 ...
30 ...
40 NEXT A
```

Dieses Programm bewirkt genau das gleiche wie das erste Beispiel, nur daß hier die Variable zunächst den Wert 10 erhält, dann 8, dann 6 usw. Wird die Angabe STEP weggelassen, so wird eine Schrittweite von 1 angenommen.

GOTO

Der Befehl GOTO bewirkt die Ausführung eines Sprungbefehls. Dabei wird hinter dem Befehl die Zeilennummer angegeben, die angesprungen werden soll. Diese Zeilennummer kann auch berechnet werden, indem eine Variable oder ein arithmetischer Ausdruck an diese Stelle geschrieben wird.

Beispiel: `10 GOTO 209`

Dieser Befehl bewirkt einen Sprung zur Zeile 209.

Beispiel: `20 GOTO 100*2+9`

Hier wird ebenfalls zur Zeile 209 gesprungen.

Es ist auch möglich, den Befehl im Direkt-Modus zu verwenden. Es wird dann zu der angegebenen Zeile gesprungen und von da an das Programm ausgeführt.

GOSUB

Mit dem GOSUB-Befehl ist es möglich, einen Unterprogramm-Aufruf durchzuführen. Dabei wird ähnlich wie beim GOTO-Befehl die Zeilennummer angegeben, die auch hier berechnet werden kann.

RETURN

Dieser Befehl stellt das Gegenstück zum GOSUB-Befehl dar. Nach Aufruf eines Unterprogramms, das mit dem RETURN-Befehl enden muß, kehrt das Programm wieder an die Stelle zurück, von der aus das Unterprogramm aufgerufen wurde.

IF

Mit diesem Befehl kann eine Entscheidung getroffen werden. IF wird von einem arithmetischen Ausdruck gefolgt. Ist der Wert ungleich 0 so wird der nachfolgende Befehl ausgeführt, andernfalls die nächste Zeile.

```
20 IF A = 2 GOTO 10
```

Wenn A den Wert 2 besitzt, wird zur Zeile 10 gesprungen. (THEN darf nicht verwendet werden).

REM

Die Anweisung REM ermöglicht es, Kommentare in das Programm einzubauen. Dabei wird der Text, der hinter einer REM-Anweisung steht, bis zum Zeilenende vom Interpreter ignoriert.

INPUT

Einen der wichtigsten BASIC-Befehle stellt der INPUT-Befehl dar. Er ermöglicht es, Daten in das Programm im Dialogverfahren einzugeben. Will man zum Beispiel in einem Programm der Variablen C einen Wert zuweisen, den der Benutzer eingeben soll, so lautet der Befehl folgendermaßen:

```
10 INPUT C
```

Bei der Ausführung des Programms druckt dann der Interpreter:

```
C:
```

Nun muß der Benutzer eine Zahl eingeben (oder einen arithmetischen Ausdruck, der dann noch berechnet wird). Will man erreichen, daß ein bestimmter Text anstatt des Variablennamens ausgedruckt wird, so gibt man diesen Text vor der Variablen in Anführungszeichen gesetzt an.

Beispiel: `10 INPUT "Geben Sie eine Zahl ein" C`

Bei Ausführung dieses Programms wird dann der angegebene Text gefolgt von einem Doppelpunkt ausgedruckt.

Es ist auch möglich, mehrere Variable einzugeben. Dazu werden sie mit Kommas getrennt.

Beispiel: `10 INPUT A,C,'Zahl'F,K`

PRINT

Mit Hilfe des PRINT-Befehls ist es möglich, Daten und Texte auszugeben. Dazu werden die verschiedenen Variablen, Zahlen und Texte mit Kommas getrennt angegeben.

Beispiel: `10 PRINT 2,B,'Text',7`

Dieses Programm bewirkt den Ausdruck der Zahl 2, dann des Inhalts der Variablen B, dann wird der Text ausgegeben und die Zahl 7. Zahlen werden mit sechs Stellen ausgegeben. Dies kann aber geändert werden. Dazu dient die sogenannte Formatanweisung. Sie kann auch mehrmals in der PRINT-Anweisung angegeben werden und bleibt bis zur nächsten Formatanweisung innerhalb einer PRINT-Anweisung wirksam. Bei der Ausführung des nächsten PRINT-Befehls in einem Programm ist wieder der Wert 6 voreingestellt. Die Formatanweisung wird mit dem Zeichen # eingeleitet und hat als Parameter eine Zahl oder einen arithmetischen Ausdruck.

Beispiel: `10 PRINT 1,#10,1,1`

Die erste 1 wird hier mit insgesamt sechs Stellen ausgedruckt, die anderen beiden mit zehn Stellen.

Wird bei der PRINT-Anweisung an die letzte Stelle ein Komma gesetzt, so wird der Zeilenvorschub unterdrückt. Der nächste Ausdruck wird dann an der letzten Position fortgesetzt.

STOP

Der STOP-Befehl beendet den Programmablauf.

CALL

Mit dem Befehl CALL ist es möglich, Unterprogramme in Maschinensprache aufzurufen. Dafür ist ein Parameter anzugeben, der die absolute Adresse des Unterprogramms angibt.

Beispiel: `10 CALLHEX(54FF)`

Dieses Programm bewirkt, daß das Maschinenprogramm auf Adresse 54FFH ausgeführt wird. Mit einem RET-Befehl (Code: C9) kann wieder in das BASIC-System zurückgekehrt werden.

OUTCHAR

Mit diesem Befehl werden Einzelzeichen ausgegeben, die auch Sonderzeichen und nicht darstellbare Zeichen sein können. Dem Befehl wird als Parameter der dezimale Wert gegeben.

Beispiel: 10 OUTCHAR(64)

Bei der Ausführung dieses Programms wird das Zeichen @ gedruckt.

OUT

Mit OUT wird einem 8080-Port direkt ein Wert zugewiesen. OUT wird dabei ähnlich wie eine Variable verwendet. Will man zum Beispiel dem PORT mit der Adresse 18H den Wert 2 zuweisen, so sieht der Befehl wie folgt aus:

10 OUT(HEX(18))=2

Mit der Funktion HEX wird hier wieder erreicht, daß der sedezimale Wert 18 in einen dezimalen Wert umgerechnet wird und dann dem OUT-Befehl zugeführt werden kann.

O$

Hierbei handelt es sich um einen speziellen Befehl, der eingeführt wurde, um auch schon in diesem kleinen BASIC-System Stringverarbeitung durchführen zu können. Der O$-Befehl ermöglicht die Ausgabe eines Textes, der auf einer beliebigen Adresse stehen kann und von 0 abgeschlossen wird (siehe auch I$, PEEK, POKE). Dazu erhält der Befehl einen zusätzlichen Parameter, nämlich die Adresse.

Beispiel: 10 O$ TOP

Hier wird ein Text ausgedruckt, der auf der ersten freien Adresse liegt und natürlich zuvor eingegeben werden mußte, z. B. mit I$ oder mit POKE.

I$

Dies ist das Gegenstück zum O$-Befehl. Dieser Befehl erhält eine Adresse als Parameter und legt dann einen Text, der eingegeben wird (ähnlich wie bei INPUT für Zahlen), auf die angegebene Adresse ab. Die Eingabe des Textes wird durch ein CR beendet. Die Länge ist über den LEN-Befehl feststellbar.

Beispiel: 10 I$ TOP

Dieses Programm legt einen Text, beginnend auf der ersten freien Adresse, ab.

POKE

POKE ist ein Befehl, mit dem Direktspeicherzugriff durchgeführt werden kann, wobei ein automatischer Schreibschutz für ein abgelegtes BASIC-Programm besteht. POKE besitzt zwei Parameter: Der erste gibt die Adresse (absolut) an, der zweite bestimmt den Wert, der auf dieser Adresse abgelegt werden soll.

Beispiel:
```
10 POKETOP + 1,5
20 POKE16000,2*5
30 POKETOP, 'T'
```

Bei Zeile 10 wird der Wert 5 (es wird nur ein Byte gespeichert, falls größere Zahlen als 255 eingegeben werden) auf die zweite freie Speicherzelle gelegt. Bei 20 wird auf die Adresse 16000 (dezimal) der Wert 10 und bei 30 auf die erste freie Adresse der ASCII-Code für den Buchstaben T abgelegt.

TAB

Mit TAB kann die aktuelle Schreibposition verändert werden. Dabei wird im Gegensatz zum Standard-BASIC der TAB-Befehl nicht in eine PRINT-Anweisung geschrieben.

Beispiel: 10 TAB(20)

Die Schreibposition wird um 20 Zeichen vorgerückt.

RND

Die Funktion RND liefert einen Zufallswert, dabei kann noch angegeben werden, in welchem Bereich dieser Wert liegen soll.

Beispiel: 10 A = RND(1000)

Die Variable A erhält einen Wert zwischen 1 und 1000.

ABS

ABS bildet den Absolutbetrag einer Zahl. ABS(–2) entspricht dem Wert 2.

SIZE

Mit SIZE kann der Speicherfreiraum ermittelt werden, der für eigene Programme noch vorhanden ist.

PEEK

Mit PEEK kann ein Direktspeicherzugriff durchgeführt werden. Dazu gibt man die Absolutadresse mit an.

Beispiel: 10 A=PEEK(HEX(2000))

A erhält den Wert des Bytes, das an der Adresse 2000 (sedezimal) steht. Im Gegensatz zu POKE wird bei PEEK ein Byte geholt.

INCHAR

Mit INCHAR kann ein Zeichen von der Konsole geholt werden. Dabei wird dieses Zeichen nicht ausgegeben. Dies ermöglicht es, Zeichen umzudefinieren oder Steuertasten zu definieren.

Beispiel: 10 B=INCHAR

HEX

Dem Befehl HEX wird in Klammern ein sedezimaler Wert gegeben, der dann in den dezimalen Wert umgerechnet wird.

IN

Mit IN kann der Wert eines 8080-Ports gelesen werden.

Beispiel: 10 A=IN(HEX(18))

Hier wird der Variablen A der Wert des Ports 18H zugewiesen.

TOP

TOP ist eine Pseudovariable. Mit dieser Funktion erhält man die Adresse des ersten freien Speicherplatzes (dezimal). Vor diesem Speicher steht das BASIC-Anwenderprogramm.

LEN

LEN ist ebenso eine Pseudovariable. Ihr Wert gibt die Länge des zuletzt mit I$ eingegebenen Textes an.

4 Weitere Eigenschaften

4.1 Variable

Als Variable stehen A bis Z zur Verfügung. Als Array (Dimension 1) wird das Zeichen ∂ verwendet, z. B.

10 ∂(10)=6

Die maximale Größe des Arrays hängt dabei von der Länge des Anwenderprogramms ab.

4.2 Arithmetik

Zahlenbereich von −32767 bis +32767; vier Grundrechenarten + * − /; Klammern können beliebig gesetzt und verschachtelt werden.

4.3 Logische Operatoren

< > <= >= # = liefern den Wert 1, falls die Aussage wahr ist, und 0, falls sie falsch ist. Die Operatoren können beliebig mit * + ... verknüpft werden.

4.4 Textoperator

Mit ' ' kann der äquivalente Sedezimalwert eines ASCII-Zeichens berechnet werden. 'A' liefert beispielsweise den Sedezimalcode 41.

4.5 Steuerzeichen

Mit verschiedenen Steuerzeichen kann man Fehler verbessern, die bei der Eingabe entstehen. CTRL C (Code 03, d. h. die Tasten „CTRL" und „C" werden gleichzeitig gedrückt) z. B. unterbricht die Ausführung eines Programms oder eines Listings. Mit CTRL A wird das zuletzt eingegebene Zeichen gelöscht (Code 01). Mit ESC (Code 1B) wird die gerade eingegebene Zeile gelöscht, wenn CR noch nicht gegeben wurde. CTRL B besitzt eine besondere Bedeutung (Code 02): Wird CTRL B ausgeführt, so gibt der Interpreter keine Zeichen mehr aus, aber empfängt noch alle Zeichen. Damit ist es möglich, Programme vom Kassettenrecorder aus einzulesen. Die Programme werden mit Hilfe des LIST-Befehls ausgegeben, während ein Kassettenrecorder mitläuft, der simultan über ein Modem an der Serienschnittstelle zum Datensichtgerät hängt. Bei der Wiedergabe wird CTRL B betätigt (zuvor NEW eingeben), dann wird der Ausgang des Kassettenmodems mit der Serienschnittstelle des Datensichtgeräteausgangs parallel geschaltet. Das Programm kann dann eingelesen werden. Am Schluß wird wieder CTRL B betätigt, so daß mit dem Interpreter wieder normal gearbeitet werden kann. Diese Unterdrückung der Ausgabe ist nötig, da sonst durch die Ausgabevorgänge eine Verlangsamung des Eingabevorgangs die Folge wäre und das System außer Tritt käme.

FSK-Modem

Wer einen Mikrocomputer ohne bereits eingebautes Kassetten-Interface besitzt, wer Funkamateur ist und an Funkfernschreiben (RTTY) interessiert ist oder wer Programme per Telefon übertragen möchte, wird um ein Modem nicht herumkommen. Die hier vorgestellten Schaltungen – Modulator und Demodulator – sind für die bei RTTY genormten Tonfrequenzen 1275 Hz und 2125 Hz ausgelegt und eignen sich für Übertragungsgeschwindigkeiten bis zu 600 Bd.

1 Der FSK-Modulator

Die Aufgabe des Modulators ist es, den beiden Logik-Pegeln L und H zwei Frequenzen zuzuordnen, hier 1275 Hz und 2125 Hz, um digitale Informationen als Tonfolgen speichern oder übertragen zu können. Dieses Verfahren nennt man *Frequency Shift Keying*, FSK also, auf gut Deutsch Frequenzumtastung. Die Schaltung in *Bild 1* benutzt ein IC vom Typ 4016, um diese beiden Töne zu erzeugen. Sie enthält vier CMOS-Analogschalter, von denen zwei als Invertierer, einer als Puffer und einer zur Umschaltung des frequenzbestimmenden Widerstandes verwendet werden. *Bild 2* zeigt eine passende Platine hierzu, *Bild 3* den Bestückungsplan.

Bild 1. Aus einem CMOS-IC 4016 läßt sich ein einfacher FSK-Modulator aufbauen

Bild 2. Die Modulator-Platine

Bild 3. Bestückungsplan der Modulator-Platine

Die Ausgangsspannung des Modulators ist rechteckförmig, der Spitze-Spitze-Wert gleicht (bei unbelastetem Ausgang) der Betriebsspannung. Für eine akustische Übertragung, z. B. in das Mikrofon eines Telefonhörers, kann am Ausgang auch ein Lautsprecher angeschaltet werden.

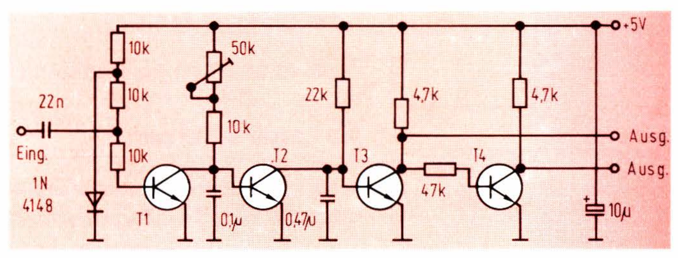
2 Der Demodulator

Um empfangsseitig jeder der beiden Tonhöhen wieder einen Logikpegel zuzuordnen, läßt sich die Schaltung nach *Bild 4* verwenden. Sie enthält einen einfachen Frequenzdiskriminator und besitzt zwei komplementäre Ausgänge. Das Eingangssignal sollte etwa zwischen 100 mV und 1 V (effektiv) liegen; besonders bei höheren Baudraten können bei zu großer oder auch zu kleiner Nf-Amplitude Fehler auftreten. *Bild 5* zeigt wiederum eine passende kleine Platine, und *Bild 6* enthält den Bestückungsplan.

3 Abgleich der Schaltungen

Beginnen wir den Abgleich am besten mit dem Modulator: Zunächst legt man an den Eingang L-Pegel an und gleicht mit dem Trimmpotentiometer „Grundfrequenz" die Ausgangsfrequenz auf 1275 Hz ab. Hier ist ein Frequenzzähler natürlich recht nützlich, eine Abweichung bis zu etwa 100 Hz ist aber zumindest in Zusammenhang mit dem beschriebenen Demodulator nicht weiter schlimm. Dann legt man den Eingang auf H und stimmt mit dem Potentiometer „Shift" auf 2125 Hz ab. Der Modulator ist damit schon betriebsbereit.

Nun verbindet man den Ausgang des Modulators über einen Spannungsteiler (10 kΩ : 1 kΩ) mit dem Eingang des Demodulators. An den Modulator-Eingang schließt man eine Rechteckspannungsquelle mit 10...300 Hz an, an einen der beiden Demodulator-Ausgänge ein Oszilloskop. Nun braucht nur noch das Demodulator-Trimmpotentiometer so eingestellt werden, daß auf dem Oszilloskop wieder die Rechteck-Eingangsfrequenz zu sehen ist.



T1...T4 = BC 108C o.ä.

Bild 4. Einfachst-Frequenzdiskriminator. Je nach angeschlossenem Gerät (RS-232, V-24, 20-mA-Stromschleife) kann der invertierte oder nichtinvertierte Ausgang verwendet werden

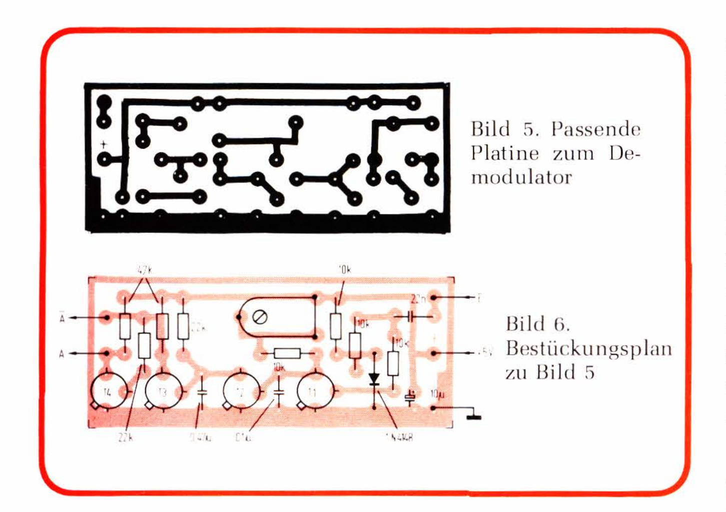


Bild 5. Passende Platine zum Demodulator

Bild 6. Bestückungsplan zu Bild 5

4 Praktische Erfahrungen

Modulator und Demodulator wurden bereits erfolgreich sowohl für Funkfernschreiben auf dem 2-m-Amateurband als auch zur Speicherung von Programmen auf Kassetten eingesetzt. Es zeigte sich, daß die mit den beiden Norm-Tonfrequenzen 1275 Hz und 2125 Hz erzielbare Baudrate bei etwa 800 liegt; 600 Bd dürften noch kein Problem darstellen, sofern man darauf achtet, daß dem Demodulator eine richtig bemessene Nf-Spannung zur Verfügung gestellt wird (s. o.). Insbesondere die Aufzeichnung von ASCII-Signalen mit einem Billigst-Kassettenrecorder zeichnet sich durch eine extrem geringe Fehlerwahrscheinlichkeit aus. Wegen seiner Unempfindlichkeit gegenüber Phasenschwankungen und schlechten Gleichlaufeigenschaften sind die Qualitätsanforderungen an den Recorder und die Bandkassette relativ gering.

Ebenso gute Erfahrungen wurden bei der ASCII-Übertragung auf dem 2-m-Amateurband in der Modulationsart FM (F2) gemacht. Kritischer wird es dagegen beim Empfang von RTTY-Stationen auf dem störungsverseuchten Kurzwellenbereich, da der Frequenzdiskriminator nicht selektiv ist. Aber was will man schon von vier Transistoren erwarten!

Herwig Feichtinger

Portable-Fernsehgerät als Monitor

Die Verwendung üblicher Fernsehgeräte als Datensichtgerät in Verbindung mit einem Terminal stellt im Vergleich zu käuflichen Monitoren eine wesentlich preisgünstigere Lösung dar. Nachfolgend wird beschrieben, welche Umbauten an einem solchen Fernsehgerät erforderlich sind.

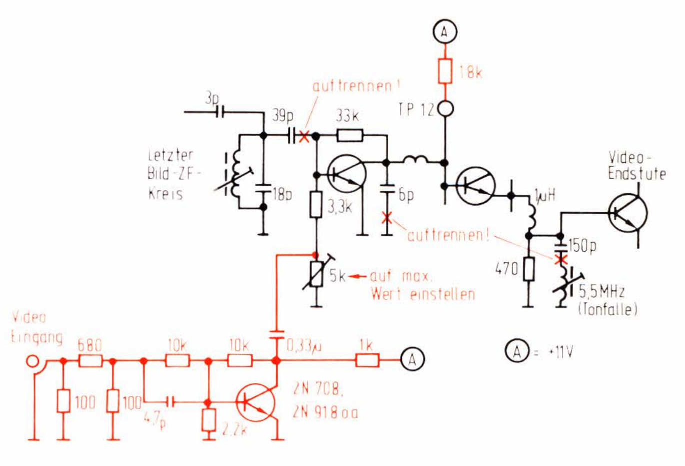
Die nachfolgenden Schaltungsangaben beziehen sich auf ein recht preiswert (ca. 200 DM) erhältliches, tragbares Schwarzweiß-Fernsehgerät für Netz- und Batteriebetrieb mit 31-cm-Bildröhre, wie es z. B. unter dem Namen „Astro-Junior", aber auch unter anderen Bezeichnungen in Kauf- und Versandhäusern zu haben ist. Das *Bild* zeigt einen Schaltungsauszug aus dem Video-Teil; der Videogleichrichter arbeitet hier aktiv mit einem Transistor, dessen Arbeitspunkt mit dem 5-kΩ-Trimmpoti normalerweise in den Kennlinien-Knick gelegt wird.

Leider ist es nicht ohne weiteres möglich, das Gerät wechselweise zum Fernsehempfang und als Terminal-Monitor zu betreiben, da die Auslegung der Schaltung sonst eine ungenügende Monitor-Videobandbreite ergeben würde. Günstiger ist es daher, VHF- und UHF-Tuner gleich ganz auszubauen und die im Bild farblich gekennzeichneten Änderungen und Erweiterungen durchzuführen. Damit läßt sich eine Videobandbreite von mehr als 10 MHz erzielen, was auch für die Darstellung von bis zu 64 Zeichen pro Zeile noch völlig ausreicht.

Der zusätzliche Transistor dient zur Phasenumkehr des Videosignals, um die richtige Polarität der Synchronimpulse herzustellen; ein 4,7-pF-Kondensator an seiner Basis korrigiert den nach oben leicht abfallenden Frequenzgang des Videoverstärkers.

Um ein Überschwingen des Videosignals zu vermeiden, ist es auch erforderlich, den 5,5-MHz-Serienschwingkreis (Tonfalle) zu unterbrechen. Ebenso wirkt es sich günstig auf hohe Frequenzen aus, wenn TP 12 über 1,8 kΩ mit der Betriebsspannung (etwa 11 V) verbunden wird.

Schließlich ist noch die Verbindung zum letzten Bild-Zf-Kreis aufzutrennen, um zu vermeiden, daß



Mit den farblich gekennzeichneten Änderungen wird aus einem Billig-Fernsehgerät ein recht brauchbarer Monitor

das Rauschen des Zf-Teils sich dem Videosignal aufmoduliert.

Der Nf-Teil des Fernsehgeräts läßt sich natürlich weiterverwenden, z. B. um die an einem I/O-Port des Mikrocomputers erzeugten Töne hörbar zu machen. Sinnvollerweise verwendet man dann die Ohrhörerbuchse als Eingang und verschaltet sie entsprechend mit dem Lautstärke-Poti des Fernsehempfängers. Als Videoeingang kann man die frühere Antennenbuchse mißbrauchen; das vom Terminal kommende Videosignal soll eine Spannung von etwa U_{ss} = 1...3 V an 75 Ω besitzen.

Will man andere Fernsehempfänger zum Monitor umbauen, so sind folgende Dinge zu beachten:

Zunächst einmal muß das Gerät galvanisch vom Netz getrennt sein. Dies ist praktisch bei allen Fernsehgeräten der Fall, die sich auch an einer Batterie bzw. einem Akku betreiben lassen, da sie einen Netztrafo enthalten, der für die Netztrennung sorgt.

Dann sollte man noch darauf achten, daß das Gerät auf der gesamten Bildfläche eine gleichbleibende Schärfe besitzt; dies erkennt man z. B. daran, ob auch in den Bildecken die Zeilen noch gut sichtbar sind. Ist dies nicht der Fall, so läßt sich das Problem auch durch Nachstellen der Fokussierspannung oft nicht beseitigen.

Eine Schwarzhalte-Schaltung ist dagegen meist entbehrlich und im oben genannten Gerät auch nicht vorhanden. Darunter versteht man eine Schaltung, die auch bei sich änderndem Bildkontrast dafür sorgt, daß Schwarz auch wirklich schwarz bleibt.

Vergessen Sie aber nicht: In jedem Fernsehempfänger kommen Spannungen bis zu einigen Kilovolt vor! Also bitte äußerste Vorsicht beim Umbau.

Herwig Feichtinger

PET-Programmiertricks

Intelligente Maschinen wie der Hobbycomputer PET haben es so an sich, daß man bei längerem Umgang mit ihnen auf Fähigkeiten stößt, von denen im dazugehörigen Handbuch nicht die Rede ist. So manche dieser Zufallsentdeckungen erweist sich, zusammen mit absichtlich gesuchten Programmiertricks, beim späteren Arbeiten mit dem Gerät als höchst nützlich. Hier ein paar Hinweise von einem Autor, der zu den ersten PET-Privatbesitzern in Deutschland zählt und nach eigenem Bekunden „schon mehr als 200 vergnügliche Hobbystunden" mit seinem Rechner verbrachte.

Gestartet werden Benutzerprogramme durch die Eingabe der Buchstabenfolge RUN, gegebenenfalls ergänzt durch die Angabe der Startadresse, und die Abschlußtaste „Return". Dieses Kommando schreibt der PET auf dem Schirm zu Protokollierungszwecken mit. Ist das Programm abgearbeitet, meldet sich das Betriebssystem mit der Mitteilung „READY" auf dem Schirm.

Es gibt nun Fälle, beispielsweise die Anfertigung von „Computerkunst" oder bei der Errichtung von Spielfeldern, wo diese Schirmausdrucke stören. Sie lassen sich mit folgendem Trick unterdrücken: Vor dem RUN-Befehl tippt man ein Fragezeichen und danach Anführungszeichen, gleichzeitig die Tasten „SHIFT" und „HOME", wieder Anführungszeichen. Nach einem Doppelpunkt zur Befehlstrennung schließt sich dann das Startkommando an. Damit findet das Programm einen unbeschriebenen Schirm vor.

Die Schluß-Meldung wird wie folgt verhindert: Als letzte Zeile wird, vor ‚END'- oder ‚STOP'-Anweisungen, ein „Scheinsprung" eingefügt, ein ‚GOTO' zu genau der Zeilennummer, in der das ‚GOTO' selbst steht. Dieser Trick unterdrückt zuverlässig die ‚READY'-Meldung, bis die Unterbrechungstaste betätigt wird.

Zwar erlaubt der PET über die PRINTTAB(N)-Anweisung den Schirmausdruck von Tabellenwerken, aber die Zahlenkolonnen haben einen Schönheitsfehler: Kommata stehen, wenn Zahlen unterschiedlicher Stellenzahl ausgegeben werden, nicht untereinander. Das beeinträchtigt die Übersichtlichkeit. Auf dem Umweg über die Verknüpfung von Zehnerlogarithmus und PRINTTAB gehts dennoch, wie das folgende Programmbeispiel zeigt, in dem die Variable TT die Zeilenposition sei, in der das Komma angeordnet ist, und X die zu druckende Zahl:

```
100 INPUT"TT,X";TT,X
110 PRINTTAB(TT-3-INT(LOG(X)/LOG(10)))X
120 GOTO100
```

Bisweilen ist es nützlich, Entscheidungen davon abhängig zu machen, ob eine bestimmte Taste gerade gedrückt ist. Das PET-Betriebssystem arbeitet mit Software-Entprellung, das heißt, gleichgültig, wie lange eine Taste gedrückt bleibt, in den Tastenspeicher wird nur einmal das der Taste zugeordnete Zeichen gelesen. Der Tastenspeicher läßt sich umgehen, wenn man mit PEEK(515) arbeitet. Es kommt bei nichtgedrückten Tasten der Wert 255 (alle 8 bit „H") zurück, bei Tastendruck der entsprechende Tastencode – und zwar so lange, wie der Finger auf der Taste bleibt. Beispiel:

```
100 PRINTPEEK(515);:GOTO100
```

Die Shifttaste ist hierbei ohne Einfluß; ihr Zustand kann über PEEK (516) gelesen werden. Ein Typ, der die Anschaffung von „Joysticks", von kleinen Spiel-Hilfskonsolen, in vielen Fällen überflüssig macht.

Der PET schreibt da auf den Schirm, wo der Cursor steht. Will man in größeren Texten oder Grafiken nur Einzelelemente ändern, so muß man erst den Cursor dorthin befördern. Das kostet Zeit und Speicherplatz – es sei denn, man bedient sich des folgenden Tricks, der einen Teil des Betriebssystems ausmanövriert (X sei der Code für das zu druckende Symbol, Z(0...24) die Nummer der Zeile, in der gedruckt werden soll, S(0...39) die Zahl der Spalte:

```
100 POKE(32768+40*Z+S),X:RETURN
```

Dieses kleine Unterprogramm schreibt blitzartig an jede beliebige Schirmstelle, ohne daß der sonstige Schirminhalt oder die Cursor-Position dabei geändert werden.

Das Blinken des Schirminhaltes ist ein auffälliges Achtung-Zeichen für den Mann am PET und kann eingesetzt werden, wenn beispielsweise ein über den IEC-BUS angeschlossenes Gerät einen Gefahrenzustand meldet oder wenn bei einem Schirmspiel eine weitere Kommunikationsmöglichkeit benötigt wird. Das Handbuch spricht nicht vom Blinken, aber der PET kann's mit Hilfe des folgenden kleinen Unterprogramms dennoch:

```
100 FORI=1TON:NEXT
110 IFF=0THENF=1:POKE59409,52
    :GOTO100
120 POKE59409,60:F=0:GOTO100
```

Hierbei bestimmt N die Blinkfrequenz. Bei N = 0 flimmert das Bild lediglich etwas, bei N = 80 ergibt sich ein Blinken etwa im Drittel-Sekunden-Takt. F hat die Funktion eines Flags.

Bedauerlicherweise verfügt der PET nicht unmittelbar über die Möglichkeit, geschlossene Programmsegmente in Bereiche mit anderen Zeilennummern am Stück zu transportieren. Man benötigt diese Fähigkeit zum Beispiel dann, wenn man beim Programmieren größere Segmente vom Band aus einer Unterprogramm-Bibliothek auslesen und in das gerade bearbeitete Programm einfügen möchte. Hier hilft folgender Trick: Die Teile der Unterprogramm-Bibliothek erhalten so hohe Zeilennummern, wie man sie üblicherweise nicht einsetzt. Zu Beginn der Eingabe liest man die Bibliothek auf übliche Weise in den PET ein, ruft sie dann, wenn man die Unterprogramme einfügen will, per LIST auf den Schirm und versieht sie, von oben beginnend, mit der neuen, „richtigen" Zeilennummer. Nach Betätigung der RETURN-Taste wird die Zeile unter der neuen Nummer eingelesen.

Vor Aufzeichnung des fertigen Programms auf Band muß die Bibliothek nur dann gelöscht werden, wenn es Speicherplatz-Probleme gibt oder auf eine kurze Programmband-Laufzeit Wert gelegt wird.

Das wär's – der Autor ist überzeugt, es lassen sich noch viele weitere solcher PET-Tricks finden, und freut sich auf entsprechende Leser-Zuschriften.

Hans-Georg Joepgen

BASIC – aber mehr als BASIC:

Die Programmiersprache des Hobbycomputers PET 2001

Zwar läßt sich der PET ohne sonderliche Schwierigkeiten in der Dezimal-Version seiner Maschinensprache oder aber, mit Hilfe eines geeigneten Übersetzungsprogramms, in MOS-Technology-Assembler programmieren. Überall da, wo keine kritischen Anforderungen an Arbeitsgeschwindigkeit oder Speicherplatzbedarf bestehen, bedient man sich jedoch mit Vorteil des fest eingespeicherten BASIC-Interpreters. Im Unterschied zu anderen wohldefinierten und maschinenunabhängigen höheren Programmiersprachen kann man bei BASIC von einer einheitlichen Sprache nicht sprechen; es existiert unterdessen eine Vielzahl von Varianten, so daß es wohl gerechtfertigt ist, einige Eigenheiten der PET-Sprache Commodore-BASIC gesondert vorzustellen.

Die „BASIC-Ursprache" ist eine schnell erlernbare und dabei erstaunlich leistungsfähige Sprache, deren Grenzen bei fortschreitender Einarbeitung aber schnell spürbar werden. Deswegen haben verschiedene Unternehmen, darunter so renommierte wie Hewlett-Packard, recht bald mit der Schaffung ihrer speziellen BASIC-Erweiterung begonnen.

Zur Zeit bemüht sich eine Kommission in den Vereinigten Staaten, festzulegen, was eigentlich noch „Grund-BASIC" ist und was Firmenspezialität. Grund dieser Bemühungen: Die Sicherstellung der Austauschbarkeit von Programmen.

Nun, Commodore kann diesen Bestrebungen gelassen entgegensehen. Der PET arbeitet sowohl Programme in der Ursprache Dartmouth-BASIC als auch, bisweilen nach einigen Änderungen bei der Zeichenketten-Verarbeitung, solche der meisten anderen verbreiteten BASIC-Spielarten ab. Darüber hinaus verfügt er über Möglichkeiten, die man bei großen Konkurrenz-Maschinen vergeblich sucht.

Da ist zuerst einmal die große Zahl der Variablen, die zur Verfügung stehen: Mit Einschluß der Integer- und String-Variablen sind es weit über 2000. Dann die Vielzahl der Befehle, die in einer einzigen Zeile zusammengefaßt werden können – sogar mehrere For-Next-Schleifen finden in ein und derselben Zeile Platz.

Ein Kapitel für sich: die komfortablen Eingabemöglichkeiten. In Commodore-BASIC müssen Anfangsvariable nicht länger in Data-Zeilen abgelegt werden, sie werden bei Bedarf per GET-Instruktion aus dem Tastenspeicher geholt, oder aber – hier zeigt sich besonders der hohe Bedienungskomfort im Umgang mit dem PET – über spezielle Input-Instruktionen, die bei Bedarf mit dem Ausdruck von Dialogsätzen auf dem Schirm gekoppelt werden können.

Zum Thema Dimensionieren von Feldern: Hier geht der PET, wenn ihm nichts anderes mitgeteilt wird, von einer zehnelementigen Dimensionierung aus, oder aber, anders betrachtet: Stößt der Rechner auf Befehle, die Feldelemente zum Gegenstand haben, und es wurde noch nicht dimensioniert, dann holt er das Versäumte nach, ohne, wie das noch immer viele seiner größeren Brüder tun, sich per Fehlermeldung zu beschweren und dann anzuhalten.

Womit wir beim Thema Fehlermeldung wären. Der BASIC-Anfänger wird die große Zahl der ausführlichen Fehlerhinweise besonders schätzen, von denen die meisten nicht nur präzise die Fehlerart, sondern auch die Zeile des ersten Auftretens mitteilen. Hauptschnitzer beim Autor übrigens: die Verwechslung von Semikolon und Doppelpunkt, Komma und Punkt, sowie leichtfertiger Umgang mit Klammern.

In einem allerdings würde der Autor dem PET noch eine Fähigkeit wünschen, die sich bei manchen Konkurrenzprodukten als nützlich erweist: automatisches Setzen von Zeilennummern beim Programmieren. Und schön wäre es auch, wenn sich Unterprogramme mit symbolischen Namen aufrufen ließen. Bei der Mehrfachverzweigung „ON X GOSUB ..." zum Beispiel will der PET die Anfangsadressen der aufzurufenden Subroutinen wissen, und das bereits zu einem Zeitpunkt, an dem sie beim Programmieren Zug um Zug noch gar nicht feststehen. Hier hilft man sich durch „Mut zur Lücke", indem man einfach auf Verdacht einige hundert Zeilennummern freiläßt. Das ist zwar weder für die Arbeitsgeschwindigkeit des Rechners noch für die Speicherausnutzung von Nachteil, aber den Systematiker, der sich ein geschlossenes Bild seines Programmausdrucks wünscht, den stört's nun mal.

Offen ist noch die Beantwortung der Frage, wieviel Zeit und Fleiß es braucht, bis der Anfänger PET-BASIC beherrscht. Dies hängt natürlich in hohem Maße von den Fähigkeiten des BASIC-Studenten ab; der folgende Hinweis gründet auf der persönlichen Erfahrung des Autors, der sich nicht zu den ‚Blitzmerkern' zählt und deswegen gerne einräumt, daß es Programmier-Talente durchaus schneller schaffen mögen. Nun, ein wenig Erfahrung aus dem Umgang mit programmierbaren Taschenrechnern und Einfachst-Mikrocomputern vorausgesetzt, ansonsten aber ungetrübt durch höhere Programmierkenntnisse: Dann muß man zum Erlernen von Commodore-BASIC etwa soviel an Lernbereitschaft und Zeit aufwenden, wie man auch zum Erwerb etwa eines Autoführerscheines braucht.

Man sieht, volles Ausschöpfen aller PET-Fähigkeiten läßt sich nicht mit der linken Hand erlernen, sondern verlangt Konzentration. Allerdings, es lohnt sich, zu erleben, wie die zu Beginn ‚tote Maschine' Hobbycomputer mit den wachsenden Fähigkeiten ihres Programmierers in steigendem Maße so etwas wie ein Eigenleben entwickelt, mehr und mehr ‚Persönlichkeit' zeigt und fast in die Rolle eines intelligenten Partners hineinwächst: Das zu erleben, gehört nach des Autors Meinung zu den befriedigendsten und reizvollsten Augenblicken, die ein technisches Steckenpferd schenken kann. *Hans-Georg Joepgen*

Literatur

1 *Gregory Yoh* (Herausgeber): „Commodore PET Computer – Users Handbook", Palo Alto, California, 1977.

2 *Julius Schärf*: „BASIC für Anfänger", fünfte Auflage, R. Oldenbourg Verlag, Wien und München, 1977.

3 *Herbert O. Ramp*: „BASIC-Praxis einschließlich Mark III-Basic", 2. Auflage, R. Oldenbourg Verlag, Wien und München, 1975.

4 *Andreas Alteneder* und *Christel Offelder*: „Basic-Praktikum", im Verlag der Siemens Aktiengesellschaft, Berlin und München, 1972.

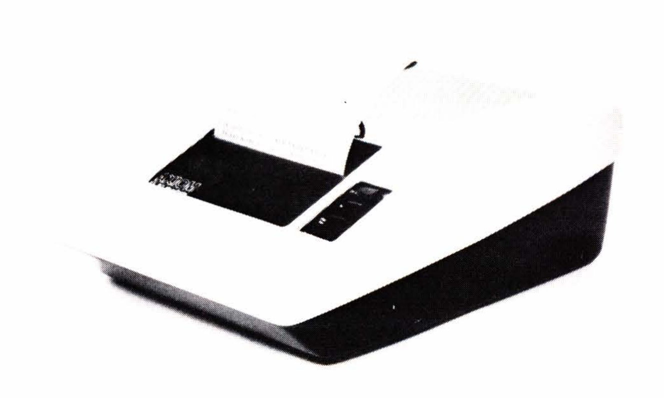
5 Commodore Büromaschinen GmbH, Neu-Isenburg (Herausgeber): „Bedienungshandbuch PET 2001", 1. Auflage (nicht deckungsgleich mit der von Yoh herausgegebenen englischsprachigen Handbuch-Version).

6 *Ralph Wells*: „PET's First Report Card", in „Kilobaud – The Small Computer Magazine", Peterborough, New Hampshire, USA, Ausgabe Mai 1978. Europäische Auslieferung durch Zeitschriftenvertrieb Monika Nedela, 7780 Markdorf.

Preiswerter Plotter ist zugleich Drucker

Das Gerät EX-820 von der Firma Axiom ist Plotter und Drucker zugleich. Grafische Darstellungen und Texte können, auch gemischt, auf Metallpapier gebracht werden. In horizontaler Richtung sind drei Auflösungsstufen bis zu 128 Punkte/Zoll wählbar. Die vertikale Auflösung ist auf 65 Punkte/Zoll festgelegt. Eine Besonderheit stellt die automatische Histogramm-Generierung dar. Balkendarstellungen können damit ohne viel Softwareaufwand realisiert werden. Auf dem 5 Zoll breiten Papier druckt das Gerät wahlweise 80, 40 oder 20 Zeichen/Zeile. Der Eingangspuffer von 512 Buchstaben kann auf über 2000 Buchstaben erweitert werden. Der Drucker/Plotter kostet 1800 Fr.

□ Vertrieb: W. Stolz AG, CH-8968 Mutschellen, Tel. (0 57) 5 46 55.



Rudolf Hofer

Biorhythmus- und Wochentagsberechnung in BASIC

Obwohl es sich beim Biorhythmus um alles andere als eine auch nur halbwegs glaubwürdige Angelegenheit handelt, hat diese Pseudo-Wissenschaft in den letzten Jahren einen unglaublichen Aufschwung erlebt. Daß sie schon über 80 Jahre alt ist, ist den wenigsten bekannt.

„Entdeckt" wurde der Biorhythmus von Dr. Fließ, einem Berliner Hals- und Nasenchirurgen, der ihn 1897 erstmals der Öffentlichkeit bekanntmachte. Dr. Fließ – der übrigens ein Jahrzehnt eng mit Sigmund Freud befreundet war – war davon überzeugt, daß das gesamte Leben zwei fundamentalen Zyklen (23 und 28 Tage) unterliegt. Später fand man noch einen weiteren von 33 Tagen. Den drei Zyklen werden die körperliche, seelische und geistige Verfassung zugeordnet. Sie beginnen mit der Geburt und laufen mit absoluter Regelmäßigkeit im Sinusformat (*Bild 1*) ab. Die „kritischen Tage" ergeben sich laut Theorie, wenn der 23- oder 28-Tage-Zyklus die Horizontallinie schneidet. Ansonsten entsprechen die Amplituden dem jeweiligen Zustand.

Der Gregorianische Kalender

Das beschriebene Programm gibt nach Eingabe des Geburtstages und des Datums, für das die Prognose gewünscht wird, die Amplitude der drei Kurven zu diesem Zeitpunkt an. Wie leicht einzusehen ist, reduziert sich das Problem auf die Berechnung der Anzahl von Tagen zwischen Geburtstag und Prognosedatum mit anschließender Berechnung der Sinuswerte.

Um das zu erreichen, muß man zunächst etwas in der Geschichte blättern: Der heute gültige Gregorianische Kalender löste 1582 den Julianischen Kalender ab, da sich darin gegenüber dem Sonnenjahr geringe Verschiebungen ergaben. Auf den 4. Oktober folgte damals gleich der 15. Allerdings wurde diese Änderung von den Protestanten teilweise erst im 18. Jahrhundert übernommen.

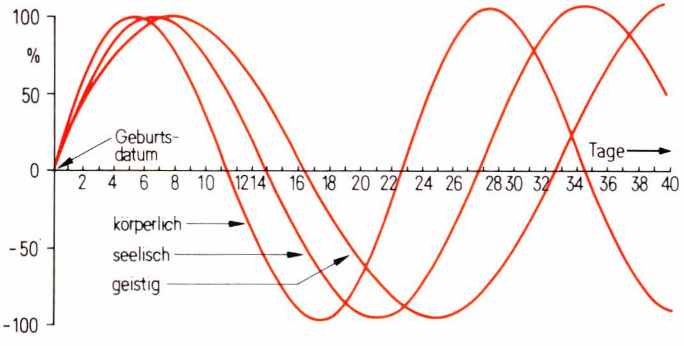


Bild 1. Prinzipieller Verlauf der Zyklen

Im Gregorianischen Kalender wird alle vier Jahre ein Schaltjahr eingefügt, das 366 Tage, also einen Tag mehr als normalerweise, dauert. Wenn die Jahreszahl durch vier teilbar ist, liegt ein Schaltjahr vor. Ausnahmen bilden diejenigen Jahre, bei denen die ersten beiden Ziffern durch vier teilbar sind, z. B. 1600, 2000, 2400 usw. Läßt man die letzte Bedingung außer Acht – wie in unserem Programm – dann sind also Berechnungen vom 1. März 1600 bis zum 28. Februar des Jahres 2000 möglich. Mit geringem Mehraufwand könnte auch diese Einschränkung aufgehoben werden – aber warum übertreiben?

Das Programm

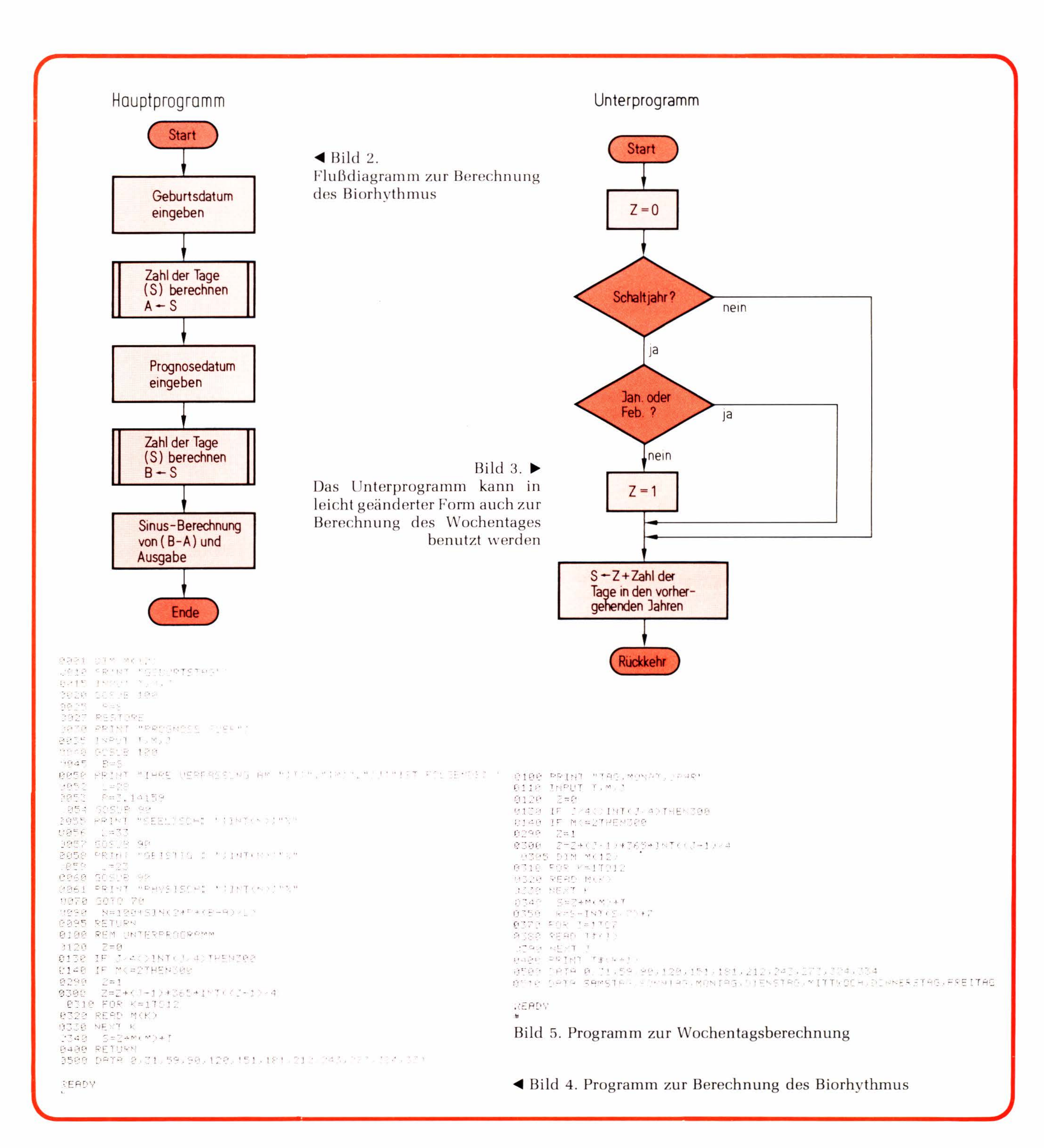
Das Hauptprogramm (*Bild 2*) nimmt Geburtsdatum und Prognosedatum entgegen und springt jeweils danach zu einem Unterprogramm, das die Anzahl der Tage von einem imaginären Zeitpunkt Null aus berechnet. Es spielt keine Rolle, wo dieser Nullpunkt liegt, da nur die Differenz zwischen Geburtstag und Prognosetag in die Berechnung eingeht.

Im Ausgabeteil (dargestellt durch das letzte Kästchen) wird noch einmal ein Unterprogramm verwendet, das aber im Flußdiagramm nicht gesondert aufgeführt ist. Es berechnet den Sinuswert der Differenz B-A (Zahl der Tage zwischen Geburt und Prognosedatum) bei einer Periode von L Tagen, wobei der Variablen L nacheinander die Dauer der drei Zyklen zugeordnet wird. Mit 100 wird multipliziert, um Prozentangaben zu erhalten.

Das eigentliche Problem wird im ersten Unterprogramm (*Bild 3*) behandelt. Es liefert als Ergebnis den Wert S, der vom erwähnten Zeitpunkt Null aus die Gesamtzahl der Tage repräsentiert. Als Hilfsgröße wird Z eingeführt – die Tageszahl der vollen Jahre vor dem betrachteten Datum. Die Formel dafür (*Bild 4*) lautet Z = (J-1)·365 + INT ((J-1)/4). J ist die Jahreszahl des betrachteten Datums, z. B. 1978. Da wir vorerst die Tage bis einschließlich 1977 wissen wollen, steht in der Formel J-1. Das wird um 365 – die Tageszahl eines normalen Jahres – multipliziert. Nun fehlen noch die Schalttage, die genau ein Viertel von J-1 ausmachen, und zwar ohne Rest, also INT ((J-1)/4) (INT = Integer = Ganzzahl).

In Zeile 300 wird zu dieser Zahl noch ein vorher definierter Wert von Z (0 oder 1) addiert. Dieser Wert ist abhängig davon, ob das betrachtete Datum in einem Schaltjahr nach dem 29. Februar liegt. Dann nämlich muß die Tageszahl um eins erhöht werden. Die entsprechende Abfrage wird in den Zeilen 130 und 140 vorgenommen. Ist die Jahreszahl nicht ohne Rest durch 4 teilbar (also $J/4 \neq INT(J/4)$), dann liegt kein Schaltjahr vor. Ist die Bedingung aber erfüllt, dann muß noch festgestellt werden, ob das Datum im Januar oder Februar liegt, also ob $M \leqq 2$ ist (M = Monat).

Um S zu erhalten, muß man jetzt noch die Anzahl der Tage im betrachteten Jahr (ohne 29. Februar, denn der ist schon berücksichtigt) ermitteln. Dazu werden den Monaten die bis zum Vormonat vergangenen Tage zugeordnet. Das geschieht in den Zeilen 310...330 und mit Hilfe der DATA-Zeile 500. Die indizierten Variablen M(1)...M(12) erhalten die Werte, 0, 31, 31 + 28, 31 + 28 + 31 usw. Der Wert M(7), der für Juli steht, ist also 181. Wer nachzählt, wird feststellen, daß bis zum Juli 181 Tage vergangen sind. Diese Zahl plus den jeweiligen Tag des Monats (T) ergibt demnach das ge-



◀ Bild 2. Flußdiagramm zur Berechnung des Biorhythmus

Bild 3. ▶ Das Unterprogramm kann in leicht geänderter Form auch zur Berechnung des Wochentages benutzt werden

```
0001 DIM M(12)
0010 PRINT "GEBURTSTAG"
0015 INPUT T,M,J
0020 GOSUB 100
0025 A=S
0027 RESTORE
0030 PRINT "PROGNOSE FUER"
0035 INPUT T,M,J
0040 GOSUB 100
0045 B=S
0050 PRINT "IHRE VERFASSUNG AM ";T;".";M;".";J;" IST FOLGENDE: "
0052 L=28
0053 P=3.14159
0054 GOSUB 90
0055 PRINT "SEELISCH: ";INT(N);"%"
0056 L=33
0057 GOSUB 90
0058 PRINT "GEISTIG : ";INT(N);"%"
0059 L=23
0060 GOSUB 90
0061 PRINT "PHYSISCH: ";INT(N);"%"
0070 GOTO 70
0090 N=100*SIN(2*P*(B-A)/L)
0095 RETURN
0100 REM UNTERPROGRAMM
0120 Z=0
0130 IF J/4<>INT(J/4)THEN300
0140 IF M<=2THEN300
0290 Z=1
0300 Z=Z+(J-1)*365+INT((J-1)/4)
0310 FOR K=1TO12
0320 READ M(K)
0330 NEXT K
0340 S=Z+M(M)+T
0400 RETURN
0500 DATA 0,31,59,90,120,151,181,212,243,273,304,334

READY
```

```
0100 PRINT "TAG,MONAT,JAHR"
0110 INPUT T,M,J
0120 Z=0
0130 IF J/4<>INT(J/4)THEN300
0140 IF M<=2THEN300
0290 Z=1
0300 Z=Z+(J-1)*365+INT((J-1)/4)
0305 DIM M(12)
0310 FOR K=1TO12
0320 READ M(K)
0330 NEXT K
0340 S=Z+M(M)+T
0350 K=S-INT(S/7)*7
0370 FOR I=1TO7
0380 READ T$(I)
0390 NEXT I
0400 PRINT T$(K+1)
0500 DATA 0,31,59,90,120,151,181,212,243,273,304,334
0510 DATA SAMSTAG,SONNTAG,MONTAG,DIENSTAG,MITTWOCH,DONNERSTAG,FREITAG

READY
```

Bild 5. Programm zur Wochentagsberechnung

◀ Bild 4. Programm zur Berechnung des Biorhythmus

wünschte Ergebnis. In Zeile 340 wird diese Berechnung durchgeführt. Die Variable M(M) ist M(1) für Januar, M(2) für Februar usw. Die Variable, die hier als Index verwendet wird, wurde gleich zu Anfang in der INPUT-Anweisung von Zeile 15 bzw. 35 eingegeben.

Drei Befehle aus dem Hauptprogramm sollten noch erwähnt werden: RESTORE in Zeile 27 sorgt dafür, daß auch beim zweiten Unterprogrammaufruf die DATA-Zeile wiederum von vorne gelesen wird. DIM M(12) in Zeile 1 erlaubt die Verwendung von zwölf Indizes statt zehn ohne DIM-Befehl, und GOTO 70 in Zeile 70 ist ein dynamischer Stopp; er dient dazu, um den Ausdruck READY am Programmende zu unterdrücken.

Berechnung des Wochentags

Mit leichten Änderungen kann das Programm auch den Wochentag eines beliebigen Datums innerbalb der eingangs erwähnten Grenzen bestimmen. Im wesentlichen wird dazu das beschriebene Unterprogramm verwendet. Es berechnet wiederum die Summe S, teilt diese durch sieben und bestimmt den Rest R (Zeile 350 in *Bild 5*) – man nennt das auch modulo 7. Auf diese Weise erhält man eine Zahl zwischen 0 und 6, die genau dem Wochentag entspricht. Nun ordnet man der String-Variablen den Index R zu und liest aus der DATA-Zeile der Reihe nach die Wochentage ein. Man muß nur einmal dafür sorgen, daß ein Datum richtig bestimmt wurde, dann stimmt auch jeder andere Tag. Daß in Zeile 400 PRINT T (R+1) steht, hat den Grund, daß im verwendeten BASIC der Index 0 verboten ist.

Beide Programme behandeln fehlerhafte Eingaben nicht gesondert. Wegen des hohen Aufwands im Verhältnis zur Programmlänge wurde darauf verzichtet.

Literatur

1 *Gardner, M.*: Mathematischer Karneval. Ullstein-Verlag.

Hans-Georg Joepgen

Mit Computerhilfe „zurück in die Vergangenheit":

PET als listenreicher Kartenspieler

Sicherlich hat sich jeder Mitbürger schon mehr als einmal gewünscht, den „Film zurückdrehen zu können", hat sich den längst vergangenen Zeitpunkt, an dem er eine später als falsch erkannte Entscheidung fällte, zurückgewünscht. Mit Computerhilfe wird dieser Wunsch jetzt ein bißchen wahr, wenngleich beschränkt auf das Geschehen in einem Kartenspiel. Das beschriebene Programm „Polterschnack" für den Hobbycomputer PET 2001 erlaubt nämlich die lückenlose Protokollierung des Verlaufs eines Kartenspiels und nach Spielende den erneuten Eintritt in jede beliebige Spielsituation – eine Art Reise in die Vergangenheit mit der Chance, vergangene Entscheidung unter präzis den gleichen Randbedingungen durch andere zu ersetzen. Vom Spielwert einmal abgesehen, veröffentlichen wir das Programm auch deswegen in ausführlicher Form, weil es in anschaulicher Weise Programmiertechniken aufzeigt, die auch auf anderen BASIC-Maschinen anwendbar sind; das Programm ist aus Software-Moduln aufgebaut, die sich ferner zum Programmieren anderer Kartenspiele eignen.

Es gab einmal eine Zeit, sie liegt fast drei Jahrzehnte zurück, da verbrachte der Autor als kurzbehoster Unterstufen-Schüler nahezu jede freie Stunde beim Kartenspiel, mit dem Spielen von „Mau-Mau". In den gleichen Monaten auch muß es gewesen sein, als den Autor die Hauptfigur eines Kunstmärchens faszinierte, der Gnom Polterschnack, der seinen Freunden Reisen in die Vergangenheit ermöglichte. Als der Verfasser jetzt nach einer Möglichkeit suchte, seine frisch angelesenen BASIC-Kenntnisse anzuwenden und die Nagelprobe mit den besonderen PET-Eigenschaften zu machen, da flossen das Märchen vom Gnom Polterschnack und ein Teil der Mau-Mau-Spielregeln zum Programm zusammen: zu einem PET-Programm, das einmal ein recht vergnügliches Spielen mit dem Computer erlaubt und das zum zweiten die Frage beantwortet: „Wie wäre das Spiel wohl ausgegangen, wenn ich vorhin statt der Karo-Dame die Karo-Sieben genommen hätte?" Kein Problem, man spult seine Datenkassette zurück, fährt wieder in die gleiche Situation, drückt dann die Taste „Z" für das Kommando „Zurück ins Spiel", und schon erlebt man, was es nun mit der vertrackten Karo-Dame in Wirklichkeit auf sich hatte.

Der Fachmann wird beim Studium der Programmliste entdecken, daß neben recht elementaren Programmiertechniken auch trickreichere Abschnitte stehen. Die Erklärung für diese vielfältigen Programmierer-Handschriften: Der Autor hat beim Anfertigen der verschiedenen Programm-Moduln, der Unterprogramme, gelernt. Nach Abschluß der Arbeit stellte er fest, daß sich mancherlei mit Hilfe der neuen Kenntnisse eleganter lösen ließe – von einer Änderung wurde dennoch Abstand genommen. Zum einen sind Speicherbedarf und Arbeitsgeschwindigkeit für

---

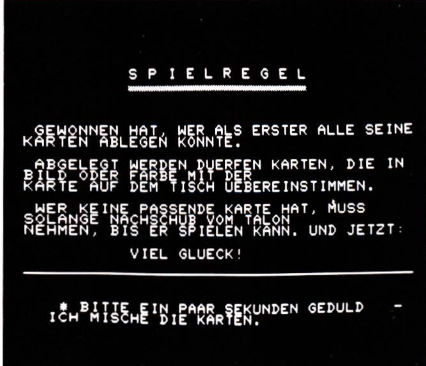
In den Programmausdrucken erscheint die Null immer mit Schrägstrich. Im Text wurde darauf generell verzichtet, da hier für die Null und den Buchstaben O ohnehin verschiedene Zeichen verwendet werden.

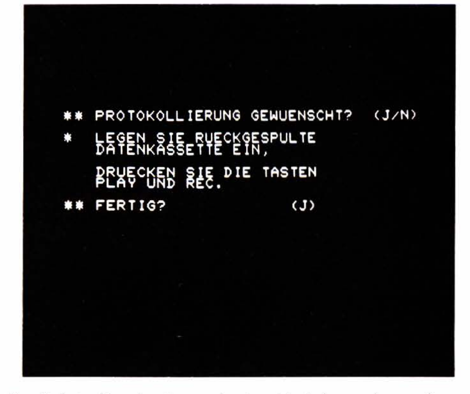
den praktischen Betrieb vollkommen ausreichend, zum zweiten hatte der Verfasser nach den letzten Programmreparaturen von Polterschnack und seinen Zeitreisen, mit Verlaub gesagt, die Nase voll: Bei Beginn der Arbeit war die Zeit nämlich gewaltig unterschätzt worden, derer es bedurfte, bis der PET „Polterschnack" endlich kapiert hatte und fehlerfrei spielte – 60 Stunden werden wohl zusammengekommen sein, und nicht immer waren es Fehler des Programmierers, die zu langen Irrwegen führten, auch das Operationssystem des PET hat seine Tücken. Hierfür ein Beispiel: In Zeile 2580 (*siehe Programmlisting*) *findet sich eine* Verzögerungsschleife bei der Bandaufzeichnung, deren Funktion erst einmal sinnlos zu sein scheint. Aufgabe: Dem PET mal Zeit zum Luftholen zu lassen, weil er sich ansonsten rettungslos verstolpert. Hier hat die zu Hilfe gerufene Firma Commodore in dankenswerter Weise geholfen, die Ursache für die chronische Arbeitsverweigerung des PET zu finden.

Nun, es ist geschafft. Der Autor hat unterdessen zwar erst mal bis auf weiteres die Lust am Kartenspielen und an Zeitreisen per Computer verloren, zu oft mußte er beim Programmumbau Testpartien durchspielen, aber Besucher im Hobby-Labor des Verfassers sind von PET und Polterschnack schier nicht wegzukriegen. Und wenn der geneigte Leser mal bei ihm zu Hause anruft und zufällig seine Frau erreicht, dann darf er sich nicht wundern, wenn sie nach den ersten Sätzen abgelenkt erscheint und merkwürdig einsilbig wird. Der PET steht nämlich bei ihm zu Hause am Telefon, und die Dame des Hauses hat sich unterdessen angewöhnt, bisweilen beim Telefonieren nebenbei eine Partie Polterschnack zu spielen.

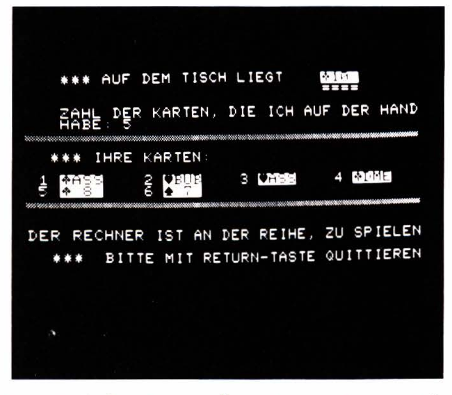
Spielverlauf und Programm

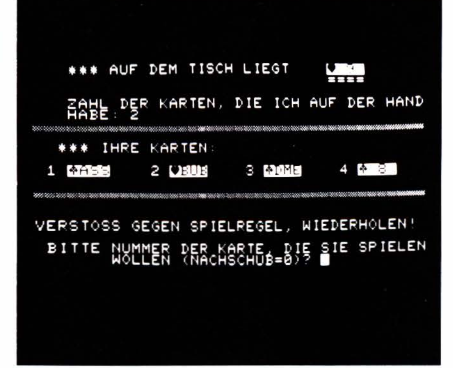


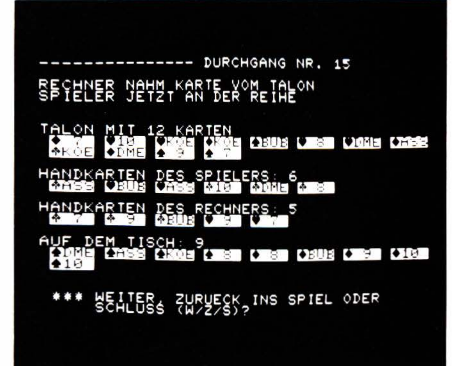




Wenn er das Programm ‚Polterschnack' in seine Speicher geladen hat, präsentiert sich der PET nicht nur als Kartenspieler mit blitzschneller Reaktion, sondern auch als Spielpartner mit Umgangsformen: Per ‚Titelblatt' begrüßt der Rechner sein menschliches Gegenüber (links). Wenn gewünscht, erläutert der PET sodann die Spielregeln (Mitte). Es folgt die Anfrage beim Spieler, ob er den Spielverlauf auf Tonband aufgezeichnet haben will: Drückt man die Taste „J" (für ‚Ja'), dann verrät der Computer, wie der Kassettenrecorder aufnahmebereit gemacht werden muß (rechts).







Das Gefecht ist in vollem Gange. Der Spieler hat eine Kreuz-Zehn auf den Tischstapel gelegt, jetzt kommt es darauf an, ob der Rechner unter seinen fünf Handkarten eine Zehn oder eine Kreuzkarte hat. Andernfalls muß er so lange Karten aufnehmen, bis er abspielen kann: So signalisiert der PET das Spielgeschehen (links). Auf Mogelversuche läßt sich der Rechner nicht ein (*Mitte*). Und so sieht das Tonband-Protokoll des Spielverlaufs aus: Alle zuvor verdeckten Karten sind jetzt zu Analysezwecken sichtbar gemacht (rechts).

Codierung der einzelnen Spielkarten

Bei allen internen Transport-, Such- und Vergleichsoperationen behandelt der PET die Spielkarten als Zahlen – ebenso beim Aufzeichnen des Spielstandes auf Band. Erst unmittelbar vor der Darstellung auf dem Bildschirm werden diese mathematischen Abbilder mit Hilfe des Übersetzungs-Unterprogramms in den Zeilen 2000...2140 in die grafischen Spielkarten-Symbole zurückverwandelt. Die Zehnerstelle gibt die Farbe, die Einerstelle das Bild der Spielkarte an. Es bedeuten im einzelnen:

1 x	Pik-Karte	Farbe
2 x	Herz-Karte	
3 x	Karo-Karte	
4 x	Kreuz-Karte	
x 1	Sieben	Bild
x 2	Acht	
x 3	Neun	
x 4	Zehn	
x 5	Bube	
x 6	Dame	
x 7	König	
x 8	As	

Beispiel: Die Zahl 37 ist das Abbild des Karo-Königs.

Die wichtigsten Flags

Der PET findet den richtigen Kurs durch die verschlungenen Wege des Programms mit Hilfe sogenannter „Flags". Das sind Variable, in denen sich der Rechner nicht-numerische Tatbestände wie z. B. einen Betrugsversuch seines menschlichen Spielpartners merkt. Bei Bedarf, das heißt an den Verzweigungsstellen des Programms, macht der PET seine Entscheidungen dann vom Status dieser Flags abhängig.

Name	Funktion		Bedeutung
F	Status-Flag	F = 1	Spieler ist an der Reihe
		F = 2	Rechner ist an der Reihe
		F = 3	Spieler nach Mogelversuch erneut dran
		F = 4	Spieler nach fehlerhafter Eingabe erneut an der Reihe
FD	Dokumentations-Flag	FD = 0	Keine Bandaufzeichnung des Spielverlaufs
		FD = 1	Aufzeichnung gewünscht
FR	Rechner-Flag	FR = 0	Rechner konnte von der Hand spielen
		FR = 1	Rechner mußte Karte vom Talon aufnehmen
FP	Prüfungs-Flag	FP = 0	Spezifizierte Karte darf nach Spielregel nicht gespielt werden
		FP = 1	Karte „paßt" an Tischkarte, sie stimmt in Farbe oder Bild mit ihr überein

Die wichtigsten Variablen

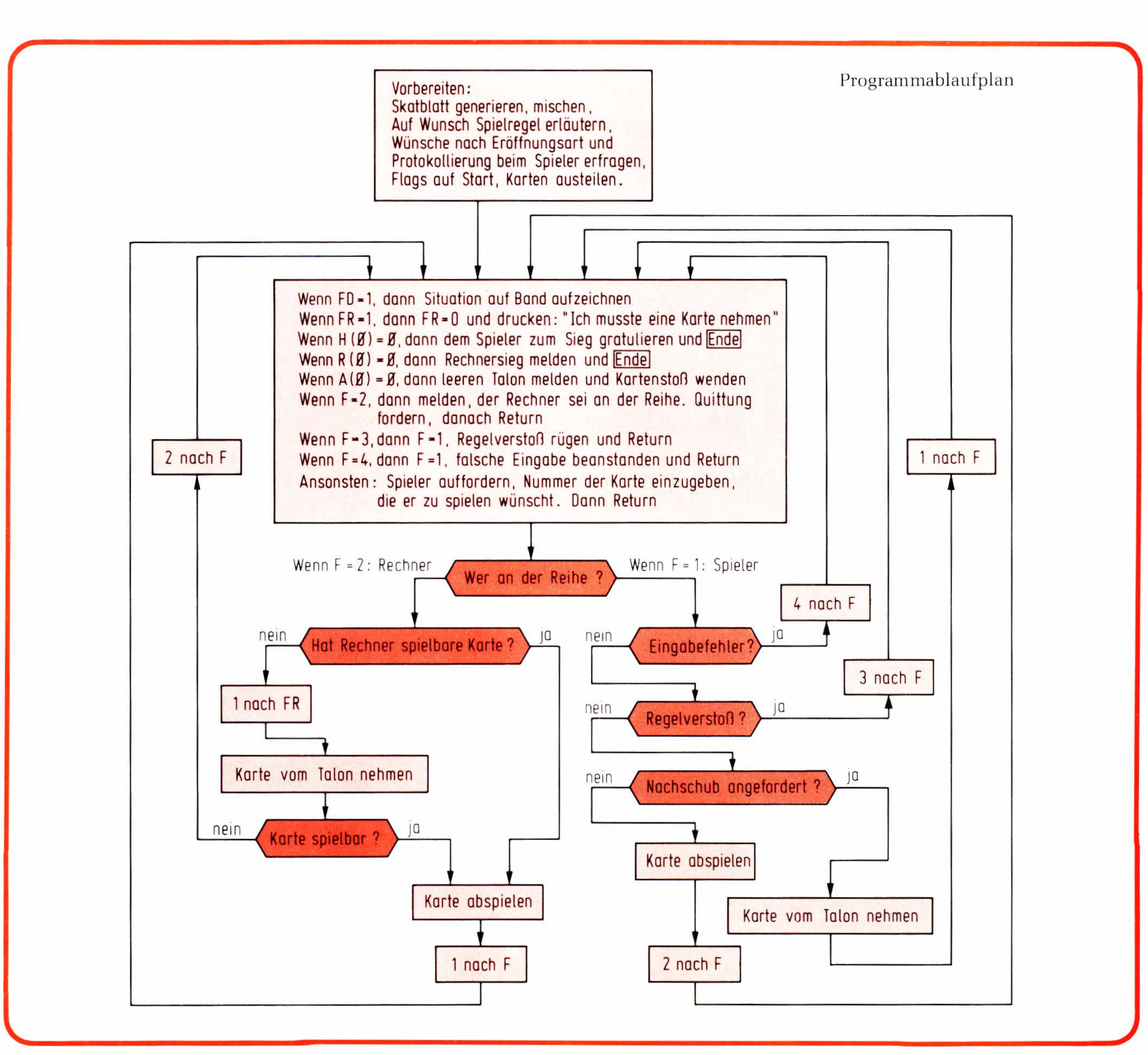
Das Verständnis des Programmlistings ist ohne eine ausführliche Variablentabelle nur mit Mühe möglich. Hier sind die Matrix-Variablen aufgeführt. Singuläre Variable entnehme man bitte, da sie in den verschiedenen Unterprogrammen teilweise verschiedene Bedeutung haben, dem Katalog der Unterprogramme.

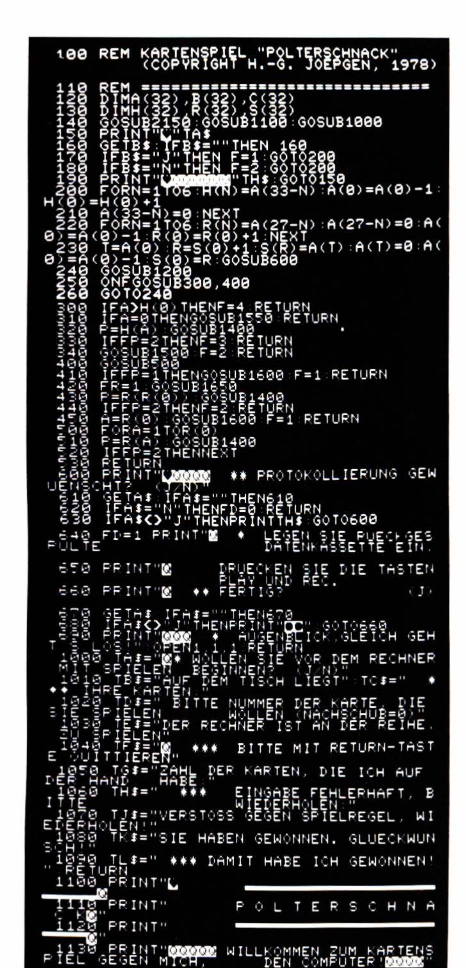
A (0...N)	Eindimensionale Matrix, die den „Talon" abbildet. Sie enthält in A (0) die Anzahl N der Talonkarten und in A (1) bis A (N) die zweistelligen Abbilder der Karten im Talon
B (0...N), C (0...N)	Eindimensionale Matrizen, die zum Mischen der Karten dienen. Organisation wie oben
H (0...N)	Matrix mit Abbildung der Handkarten des Spielers. In H (0) wieder die Anzahl
R (0...N)	Handkarten des Rechners. Anordnung wie oben
S (0...N)	Abbildung des Kartenstapels auf dem Tisch: Zu Spielbeginn enthält die Matrix s in S (0) die Zahl 1 und in S (1) das Abbild der 13. Talonkarte. Die ersten zwölf Talonkarten wurden zu Beginn als Handkarten verteilt. Im Verlauf des Spiels füllt sich die Matrix S. Ist der Talon aufgebraucht, werden alle Stapelkarten bis auf die letzte, die auf dem Tisch liegenbleibt, zum neuen Talon („Stoß wenden")

Katalog der Unterprogramme

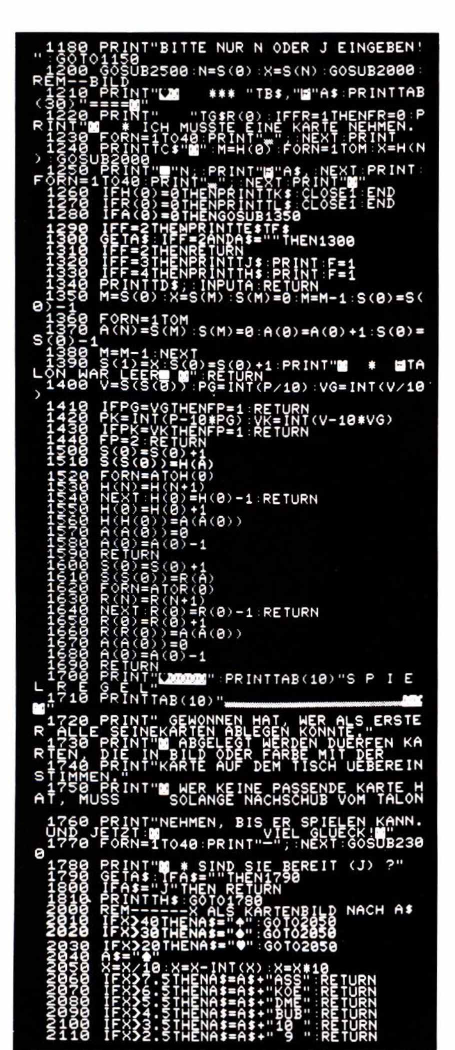
Anf.-zeile	Funktion
300	„Spieler-Spiel": Übernimmt in A die Nummer der Karte, die der Spieler ablegen will, prüft auf Irrtum oder Regelverstoß und führt, wenn zulässig, das Abspielen der Karte durch
400	„Rechner-Spiel": Veranlaßt den Rechner, seine Karten nach einer spielbaren abzusuchen. Wenn erfolgreich, erfolgt Kartentransfer auf den Tisch
500	Unterprogramm zu 400: Macht die Kleinarbeit für obiges Subsegment, übernimmt nacheinander jede Rechnerkarte in P und übergibt sie zur Prüfung dem Unterprogramm 1400. Rückmeldung des Ergebnisses in FP
600	Fragt den Spieler, ob er Protokollierung auf Tonband wünsche, setzt in diesem Fall FD auf 1, gibt Anweisungen zur Bedienung des Recorders und bereitet Aufzeichnung vor
1000	Dialog-Sätze werden in String-Variable eingelesen. Eine Art Wörterbuch des Rechners für Mitteilungen in deutscher Umgangssprache
1100	„Titelblatt": Veranlaßt das Erscheinen des Anfangstextes auf dem Schirm, fragt an, ob die Spielregeln bereits bekannt sind, und erläutert diese erforderlichenfalls
1200	„Spielstand": Zeichnet auf dem Schirm die Handkarten des Spielers sowie die oberste Stoßkarte, teilt Randbedingungen des Spielgeschehens mit, tadelt nach Mogelversuch, prüft auf Spielende und fordert den Spieler zu den erforderlichen Aktivitäten auf. Aktualisiert alle Flags

1350	„Stoß wenden“: Macht alle Tischkarten bis auf die letzte, die liegenbleibt, zum neuen Talon. X: letzte S-Karte
1400	Übernimmt Kartenbild in P und teilt in FP mit, ob P an die Tischkarte S(S(0)) nach der Spielregel angelegt werden darf. V: Vergleichskarte. PK, PC, VK, VG: Prüfvariable
1500	„Transfer“: Die mit ihrer Ordnungszahl in A bezeichnete Karte wird von den Handkarten des Spielers auf den Tisch transferiert
1550	Kartentransport der obersten Talonkarte in die Handkarten des Spielers („eine Karte aufnehmen“)
1600	Die mit ihrer Ordnungszahl in A spezifierte Rechner-Handkarte wird auf den Tisch befördert
1650	Der Rechner nimmt eine Karte vom Talon
1700	„Spielregel-Tafel“: Erklärt Spielregel und wartet, bis der Spieler kapiert und quittiert hat
2000	„Übersetzen“: Übernimmt in X die Doppelziffer für die Codebezeichnung einer Karte und antwortet in A-String mit der Zeichnung der Karte
2150	„Skatblatt generieren“: Erstellt die Ziffern-Abbilder der 32 Skatkarten und füllt die Matrix A, den Talon, damit
2300	„Mischen“: Der geordnete Talon wird Karte um Karte in die Matrizen B und C umsortiert. Dabei entscheidet jeweils ein Zufallsgenerator, ob eine Karte nach B oder C kommt. B wird von unten, C von oben gefüllt. Danach kommen die nun in ungeordneter Reihenfolge liegenden Karten zurück in die Matrix A. Das ganze geschieht viermal. Unterdessen wird der Spieler um ein paar Sekunden Geduld gebeten: „Ich mische die Karten“, schreibt der Rechner auf den Schirm
2500	Wenn FD = 1, wird der Spielstand mitsamt der Kartenverteilung und allen Randbedingungen zur späteren Wiedergabe auf Band aufzeichnet
3000	„Wiedergabe“: Dieses Programmsegment bildet nach Abschluß des Spieles ein eigenständiges Hauptprogramm. Nach dem Rückspulen der Datenkassette wird es mit dem Tastenbefehl RUN 3000 gestartet und gibt dann, Durchgang um Durchgang, den Spielverlauf wieder – diesmal mit offenen Karten. Nach jedem Durchgang wird der Spieler gefragt, ob er die Wiedergabe fortsetzen, das Programm beenden oder in die Spielsituation mit verdeckten Karten wiedereintreten möchte. Der Wunsch wird prompt erfüllt

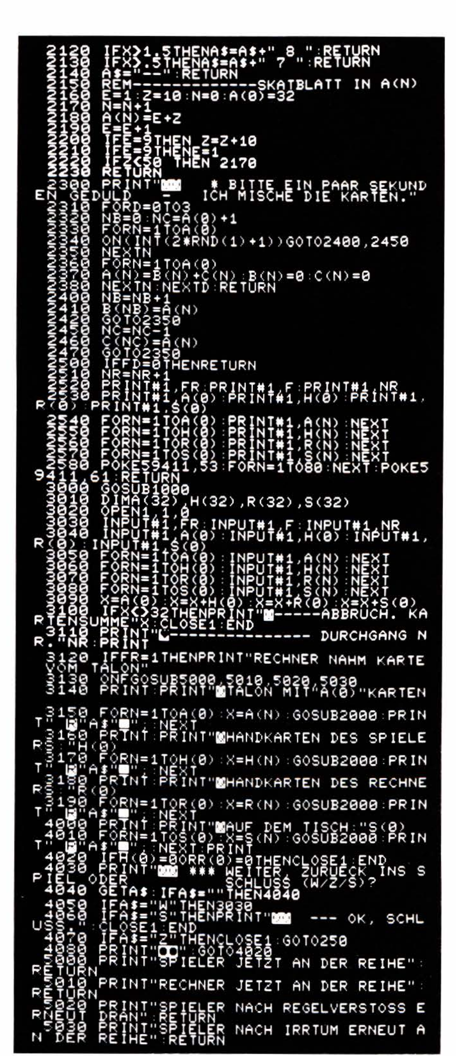




```
100 REM KARTENSPIEL "POLTERSCHNACK"
        (COPYRIGHT H.-G. JOEPGEN, 1978)
110 REM ================================
120 DIMA(32),B(32),C(32)
130 DIMH(32),R(32),S(32)
140 GOSUB2150:GOSUB1100:GOSUB1000
150 PRINT""TA$
160 GETB$:IFB$=""THEN 160
170 IFB$="J"THEN F=1:GOTO200
180 IFB$="N"THEN F=2:GOTO200
190 PRINT""TH$:GOTO150
200 FORN=1TO6:H(N)=A(33-N):A(0)=A(0)-1:H(0)=H(0)+1
210 A(33-N)=0:NEXT
220 FORN=1TO6:R(N)=A(27-N):A(27-N)=0:A(0)=A(0)-1:R(0)=R(0)+1:NEXT
230 T=A(0):R=S(0)+1:S(R)=A(T):A(T)=0:A(0)=A(0)-1:S(0)=R:GOSUB600
240 GOSUB1200
250 ONFGOSUB300,400
260 GOTO240
300 IFA>H(0)THENF=4:RETURN
310 IFA=0THENGOSUB1550:RETURN
320 P=H(A):GOSUB1400
330 IFFP=2THENF=3:RETURN
340 GOSUB1500:F=2:RETURN
400 GOSUB500
410 IFFP=1THENGOSUB1600:F=1:RETURN
420 FR=1:GOSUB1650
430 P=R(R(0)):GOSUB1400
440 IFFP=2THENF=2:RETURN
450 A=R(0):GOSUB1600:F=1:RETURN
500 FORA=1TOR(0)
510 P=R(A):GOSUB1400
520 IFFP=2THENNEXT
530 RETURN
600 PRINT"  ** PROTOKOLLIERUNG GEWUENSCHT?  (J/N)"
610 GETA$:IFA$=""THEN610
620 IFA$="N"THENFD=0:RETURN
630 IFA$<>"J"THENPRINTTH$:GOTO600
640 FD=1:PRINT"  *  LEGEN SIE BUECHGESPULTE        DATENKASSETTE EIN."
650 PRINT"     DRUECKEN SIE DIE TASTEN
                 PLAY UND REC.
660 PRINT"  ** FERTIG?          (J)
670 GETA$:IFA$=""THEN670
680 IFA$<>"J"THENPRINT"":GOTO660
690 PRINT"  *  AUGENBLICK,GLEICH GEHT'S LOS!":OPEN1,1,1:RETURN
1000 TA$="* WOLLEN SIE VOR DEM RECHNER MIT SPIELEN  BEGINNEN?  (J/N)"
1010 TB$="AUF DEM TISCH LIEGT":TC$=" *** IHRE KARTEN."
1020 TD$=" BITTE NUMMER DER KARTE, DIE SIE SPIELEN      WOLLEN (NACHSCHUB=0)"
1030 TE$="DER RECHNER IST AN DER REIHE. ZU SPIELEN"
1040 TF$="  ***  BITTE MIT RETURN-TASTE QUITTIEREN"
1050 TG$="ZAHL DER KARTEN, DIE ICH AUF DER HAND  HABE."
1060 TH$="  ***   EINGABE FEHLERHAFT, BITTE           WIEDERHOLEN."
1070 TJ$="VERSTOSS GEGEN SPIELREGEL, WIEDERHOLEN!"
1080 TK$="SIE HABEN GEWONNEN. GLUECKWUNSCH!"
1090 TL$=" *** DAMIT HABE ICH GEWONNEN!":RETURN
1100 PRINT"
1110 PRINT"        P O L T E R S C H N A C K"
1120 PRINT"
1130 PRINT" WILLKOMMEN ZUM KARTENSPIEL GEGEN MICH,       DEN COMPUTER!"
1140 PRINT" *** KENNEN SIE DIE SPIELREGELN BEREITS?":PRINT"       (J/N)
BREAK
1150 GETA$:IFA$=""THEN1150
1160 IFA$="N"THENGOSUB1700:RETURN
1170 IFA$="J"THENGOSUB2300:RETURN
```



```
1180 PRINT"BITTE NUR N ODER J EINGEBEN!":GOTO1150
1200 GOSUB2500:N=S(0):X=S(N):GOSUB2000:REM--BILD
1210 PRINT"   *** "TB$,""A$:PRINTTAB(30)"===="
1220 PRINT"   "TG$R(0):IFFR=1THENFR=0:PRINT"  * ICH MUSSTE EINE KARTE NEHMEN.
1230 FORN=1TO40:PRINT"";:NEXT:PRINT
1240 PRINTTC$"":M=H(0):FORN=1TOM:X=H(N):GOSUB2000
1250 PRINT""N;:PRINT""A$,:NEXT:PRINT:FORN=1TO40:PRINT"";:NEXT:PRINT""
1260 IFH(0)=0THENPRINTTK$:CLOSE1:END
1270 IFR(0)=0THENPRINTTL$:CLOSE1:END
1280 IFA(0)=0THENGOSUB1350
1290 IFF=2THENPRINTTE$TF$
1300 GETA$:IFF=2ANDA$=""THEN1300
1310 IFF=2THENRETURN
1320 IFF=3THENPRINTTJ$:PRINT:F=1
1330 IFF=4THENPRINTTH$:PRINT:F=1
1340 PRINTTD$;:INPUTA:RETURN
1350 M=S(0):X=S(M):S(M)=0:M=M-1:S(0)=S(0)-1
1360 FORN=1TOM
1370 A(N)=S(M):S(M)=0:A(0)=A(0)+1:S(0)=S(0)-1
1380 M=M-1:NEXT
1390 S(1)=X:S(0)=S(0)+1:PRINT"  *  TALON WAR LEER ":RETURN
1400 V=S(S(0)):PG=INT(P/10):VG=INT(V/10)
1410 IFPG=VGTHENFP=1:RETURN
1420 PK=INT(P-10*PG):VK=INT(V-10*VG)
1430 IFPK=VKTHENFP=1:RETURN
1440 FP=2:RETURN
1500 S(0)=S(0)+1
1510 S(S(0))=H(A)
1520 FORN=ATOH(0)
1530 H(N)=H(N+1)
1540 NEXT:H(0)=H(0)-1:RETURN
1550 H(0)=H(0)+1
1560 H(H(0))=A(A(0))
1570 A(A(0))=0
1580 A(0)=A(0)-1
1590 RETURN
1600 S(0)=S(0)+1
1610 S(S(0))=R(A)
1620 FORN=ATOR(0)
1630 R(N)=R(N+1)
1640 NEXT:R(0)=R(0)-1:RETURN
1650 R(0)=R(0)+1
1660 R(R(0))=A(A(0))
1670 A(A(0))=0
1680 A(0)=A(0)-1
1690 RETURN
1700 PRINT"":PRINTTAB(10)"S P I E L R E G E L"
1710 PRINTTAB(10)""
1720 PRINT" GEWONNEN HAT, WER ALS ERSTER ALLE SEINEKARTEN ABLEGEN KONNTE."
1730 PRINT" ABGELEGT WERDEN DUERFEN KARTEN, DIE IN BILD ODER FARBE MIT DER
1740 PRINT"KARTE AUF DEM TISCH UEBEREINSTIMMEN."
1750 PRINT" WER KEINE PASSENDE KARTE HAT, MUSS     SOLANGE NACHSCHUB VOM TALON
1760 PRINT"NEHMEN, BIS ER SPIELEN KANN. UND JETZT:        VIEL GLUECK!"
1770 FORN=1TO40:PRINT"-";:NEXT:GOSUB2300
1780 PRINT" * SIND SIE BEREIT (J) ?"
1790 GETA$:IFA$=""THEN1790
1800 IFA$="J"THEN RETURN
1810 PRINTTH$:GOTO1780
2000 REM------X ALS KARTENBILD NACH A$
2010 IFX>40THENA$="♠":GOTO2050
2020 IFX>30THENA$="♦":GOTO2050
2030 IFX>20THENA$="♥":GOTO2050
2040 A$="♣"
2050 X=X/10:X=X-INT(X):X=X*10
2060 IFX>7.5THENA$=A$+"ASS":RETURN
2070 IFX>6.5THENA$=A$+"KOE":RETURN
2080 IFX>5.5THENA$=A$+"DME":RETURN
2090 IFX>4.5THENA$=A$+"BUB":RETURN
2100 IFX>3.5THENA$=A$+"10 ":RETURN
2110 IFX>2.5THENA$=A$+" 9 ":RETURN
```



```
2120 IFX>1.5THENA$=A$+" 8 ":RETURN
2130 IFX>.5THENA$=A$+" 7 ":RETURN
2140 A$="--":RETURN
2150 REM-------------SKATBLATT IN A(N)
2160 E=1:Z=10:N=0:A(0)=32
2170 N=N+1
2180 A(N)=E+Z
2190 E=E+1
2200 IFE=9THEN Z=Z+10
2210 IFE=9THENE=1
2220 IFZ<50 THEN 2170
2230 RETURN
2300 PRINT"  * BITTE EIN PAAR SEKUNDEN GEDULD  -  ICH MISCHE DIE KARTEN."
2310 FORD=0TO3
2320 NB=0:NC=A(0)+1
2330 FORN=1TOA(0)
2340 ON(INT(2*RND(1)+1))GOTO2400,2450
2350 NEXTN
2360 FORN=1TOA(0)
2370 A(N)=B(N)+C(N):B(N)=0:C(N)=0
2380 NEXTN:NEXTD:RETURN
2400 NB=NB+1
2410 B(NB)=A(N)
2420 GOTO2350
2450 NC=NC-1
2460 C(NC)=A(N)
2470 GOTO2350
2500 IFFD=0THENRETURN
2510 NR=NR+1
2520 PRINT#1,FR:PRINT#1,F:PRINT#1,NR
2530 PRINT#1,A(0):PRINT#1,H(0):PRINT#1,R(0):PRINT#1,S(0)
2540 FORN=1TOA(0):PRINT#1,A(N):NEXT
2550 FORN=1TOH(0):PRINT#1,H(N):NEXT
2560 FORN=1TOR(0):PRINT#1,R(N):NEXT
2570 FORN=1TOS(0):PRINT#1,S(N):NEXT
2580 POKE59411,53:FORN=1TO80:NEXT:POKE59411,61:RETURN
3000 GOSUB1000
3010 DIMA(32),H(32),R(32),S(32)
3020 OPEN1,1,0
3030 INPUT#1,FR:INPUT#1,F:INPUT#1,NR
3040 INPUT#1,A(0):INPUT#1,H(0):INPUT#1,R(0):INPUT#1,S(0)
3050 FORN=1TOA(0):INPUT#1,A(N):NEXT
3060 FORN=1TOH(0):INPUT#1,H(N):NEXT
3070 FORN=1TOR(0):INPUT#1,R(N):NEXT
3080 FORN=1TOS(0):INPUT#1,S(N):NEXT
3090 X=A(0):X=X+H(0):X=X+R(0):X=X+S(0)
3100 IFX<>32THENPRINT"-----ABBRUCH. KARTENSUMME"X:CLOSE1:END
3110 PRINT"--------------- DURCHGANG NR."NR:PRINT
3120 IFFR=1THENPRINT"RECHNER NAHM KARTE VOM TALON"
3130 ONFGOSUB5000,5010,5020,5030
3140 PRINT:PRINT"TALON MIT"A(0)"KARTEN
3150 FORN=1TOA(0):X=A(N):GOSUB2000:PRINT" "A$" ";:NEXT
3160 PRINT:PRINT"HANDKARTEN DES SPIELERS:"H(0)
3170 FORN=1TOH(0):X=H(N):GOSUB2000:PRINT" "A$" ";:NEXT
3180 PRINT:PRINT"HANDKARTEN DES RECHNERS:"R(0)
3190 FORN=1TOR(0):X=R(N):GOSUB2000:PRINT" "A$" ";:NEXT
4000 PRINT:PRINT"AUF DEM TISCH:"S(0)
4010 FORN=1TOS(0):X=S(N):GOSUB2000:PRINT" "A$" ";:NEXT:PRINT
4020 IFH(0)=0ORR(0)=0THENCLOSE1:END
4030 PRINT" *** WEITER, ZURUECK INS SPIEL ODER        SCHLUSS (W/Z/S)?
4040 GETA$:IFA$=""THEN4040
4050 IFA$="W"THEN3030
4060 IFA$="S"THENPRINT"  --- OK, SCHLUSS.":CLOSE1:END
4070 IFA$="Z"THENCLOSE1:GOTO250
4080 PRINT"":GOTO4020
5000 PRINT"SPIELER JETZT AN DER REIHE":RETURN
5010 PRINT"RECHNER JETZT AN DER REIHE":RETURN
5020 PRINT"SPIELER NACH REGELVERSTOSS ERNEUT DRAN":RETURN
5030 PRINT"SPIELER NACH IRRTUM ERNEUT AN DER REIHE":RETURN
```

Das sind die Bytes und Bits, die den PET zum perfekten Kartenspieler machen. Zur Analyse des Programms sei empfohlen, die Tabellen für Variable und Unterprogramme sowie den Ablaufplan zu Rate zu ziehen. Zur Strategie des Rechners ist zu sagen, daß er fair spielt: Er schaut dem Gegner niemals in die Karten, schnüffelt nicht im Talon und benimmt sich auch sonst anständig. Grobe Schnitzer wie das Spielen unzulässiger Karten leistet sich der Rechner nicht. Dies, und neben der Geschwindigkeit seiner Reaktionen allein dies, hat er seinem menschlichen Gegenüber voraus.

The First Book Of KIM

Vertrieb in Deutschland: Human Electron GmbH, Mutzenreisstraße 50, 7302 Ostfildern 2; auch zu haben bei: ABC-Computershop, Schellingstraße 33, 8000 München 40; Preis: DM 20.–.

Der von MOS Technology hergestellte KIM-1 gehört heute zu den am weitesten verbreiteten Hobby- und Übungs-Mikrocomputern. Kein Wunder, daß für diesen Single-Board-Computer heute ein Programm-Angebot zur Verfügung steht, von dem manche Prozessor-Hersteller nur träumen können. „The First Book Of KIM" ist eine Programmsammlung für alle jene, die den KIM-1 im „Rohzustand", also ohne periphere Geräte und nur mit dem auf der Platine enthaltenen Hexadezimal-Tastenfeld nebst Siebensegment-Display betreiben.

Wenn das 175 Seiten umfassende Buch bisher auch leider nur in englischer Sprache zur Verfügung steht, so dürfte es doch für die meisten KIM-Besitzer interessant sein. Es ist in sechs Teile gegliedert: Am Anfang steht eine Einführung in Programmiertechniken für Anfänger; es folgt eine Sammlung von Spielprogrammen, dann eine Reihe recht nützlicher Programmier-Hilfen (darunter auch ein „Hypertape"-Programm, das sich für Bandaufzeichnung mit hoher Geschwindigkeit eignet, ohne Hardware-Änderungen am KIM durchführen zu müssen). Ein Kapitel beschäftigt sich mit der Speicher- und Peripherie-Expansion, ein weiteres beschreibt die Verwendung der I/O-Leitungen. Der Anhang ist schließlich ein buntes Potpourri nützlicher Hinweise für Betrieb und Programmierung des kleinen Elektronenhirns. *Fe.*

BASIC Software Library

Band I (24.95 $), Band II (24.95 $), Band III (39.95 $), Band IV (9.95 $), Band V (9.95 $). Scientific Research, 1712 Farmington Court, Crofton MD 21114, USA.

Die fünf Bände der „BASIC Software Library" enthalten eine umfangreiche Sammlung von BASIC-Programmen, nach verschiedenen Sachgebieten geordnet. Am interessantesten für den technisch orientierten Leser ist Band II, der zahlreiche mathematische Probleme behandelt. Einige Beispiele: Interpolation, Fourier-Transformation, Tiefpaßfilter-Berechnung, Dreiecksberechnung, Bestimmung der Position von Sonne und Mond. Daneben enthält er zahlreiche statistische Programme und Plott-Routinen für den Ausdruck verschiedener Funktionen. Die anderen Bände widmen sich vor allem den Themen „Buchhaltung" und „Spiele".

Obwohl sich die BASIC-Programme zum größten Teil selbst erklären, ist jedem ein kurzer Text mit Erläuterungen vorangestellt. Als besonders nützlich erweist sich die Angabe des benötigten Speicherbereichs und der Grenzen des Programms. Erwähnenswert ist noch, daß überwiegend Standard-BASIC verwendet wurde. Es werden also keine besonderen Anforderungen an den ausführenden Computer gestellt. *Ho*

Schönes vom Schirm:

„Grafikgenerator" macht den PET zum Künstler

Wenn man einige Zeichen in den Hobbycomputer PET eingibt und ihn veranlaßt, diese Zeichen fortlaufend und ohne Unterbrechung aneinanderzureihen, so entstehen reizvolle grafische Muster. Ein Einfachprogramm, mit dem dies schon sehr schön gelingt, besteht nur aus zwei Zeilen:

```
10 INPUT A$
20 PRINT A$; : GOTO 20
```

Dieses Miniprogramm ist sekundenschnell eingetastet, weist aber einige Mängel auf. So läßt sich das Erscheinen von „Prompting-Texten" wie „Ready" in der Schirmgrafik nicht vermeiden, die Produktion immer weiterer Zeilen auf dem Schirm kann nur durch einen STOP-Befehl vom Tastenfeld her verhindert werden, und die Verwendung der als Grafikelemente besonders geeigneten Zeichen „Schwarz in weißem Feld" ist nicht möglich. Das im folgenden erläuterte Programm vermeidet diese Unzulänglichkeiten und bietet neben der Möglichkeit zu ungeschmälerter Nutzung des gesamten Zeichenvorrates auch einigen Bedienungskomfort.

Soll der beliebige Wechsel von Positiv und Negativ-Zeichen möglich sein, kann der „Input-Befehl" zum Einlesen des Grundmusters nicht verwendet werden. Man muß auf die „Get-Operation" ausweichen, was erst einmal den Nachteil hat, daß die Eingabe nicht auf dem Schirm mitgeschrieben wird. Abhilfe schafft eine Druckroutine, die allerdings selektiv wirken muß, weil der Rechner sonst das Ausdrucken gewisser Sonderzeichen als Systembefehle interpretiert.

Schließlich muß noch sichergestellt werden, daß der Rechner nach dem Vollschreiben des Schirms in einer Warteschleife verharrt, ohne sich zu melden und damit die Grafik zu zerstören. Zur Anfertigung eines neuen Musters muß die Warteschleife aber beliebig verlaßbar sein – dazu soll eine Taste definiert werden, die man nicht zur Generierung des Grundmusters be-

```
GRAFIK-GENERATOR

MIT HILFE DIESES PROGRAMMS ENTWIRFT
DER PET STRUKTUREN, DIE ER AUS
VON IHNEN EINGEGEBENEN GRUNDMUSTERN
ZUSAMMENSETZT.

KORREKTUREN: RETURN-TASTE.

EBENSO,WENN SIE NACH FERTIGSTELLUNG
DER GRAFIK NEU EINGEBEN WOLLEN.

BEENDEN SIE IHRE EINGABE JEWEILS MIT
DER  CLR-TASTE.
-----------------------------------

WENN SIE BEGINNEN WOLLEN, DRUECKEN
SIE BITTE JETZT DIE LEERTASTE (SPACE).
```

So stellt sich das beschriebene Programm vor

nötigt. Das sind die Anforderungen an ein solches Programm. Wie sie erfüllt wurden, das soll nachfolgend gezeigt werden. Dazu wird ein Blick in die Programm-Auflistung (*Bild 1*) empfohlen: Das Programm beginnt mit Anweisungen an den Benutzer in der Gruppe „Titelblatt" (Zeilen 100...170) und wartet dann das Startkommando ab (Zeilen 175...180). Soll das Programm vor Fehlbedienung durch ungeübte Benutzer geschützt werden, kann in Zeile 190 ein „STOP-Befehl" eingefügt werden: dafür ist Raum freigelassen worden.

Das Programmsegment „Einlesen des Grundmusters" in den Zeilen 200...250 löscht zu Beginn den Schirm, setzt alle Variable auf Null und wartet in Zeile 210 auf Eingaben. Erfolgt das Kommando zum Abschluß der Eingabe durch Drücken der Taste „CLR HOME", fügt der Rechner dem in B-String gesammelten Zeichenvorrat ein Zeichen „Rückschalten von Re-



```
 100 PRINT"■          GRAFIK-GENERATOR
 105 PRINT"
 110 REM  (COPYRIGHT H.-G. JOEPGEN, 1978
)
 115 PRINT"■
 120 PRINT"   MIT HILFE DIESES PROGRAMMS
ENTWIRFT
 125 PRINT"   DER PET STRUKTUREN, DIE ER
AUS
 130 PRINT"   VON IHNEN EINGEGEBENEN GRU
NDMUSTERN
 135 PRINT"   ZUSAMMENSETZT.
 140 PRINT"■   KORREKTUREN: RETURN-TASTE
.
 145 PRINT"■   EBENSO,WENN SIE NACH FERT
IGSTELLUNG
 150 PRINT"   DER GRAFIK NEU EINGEBEN WO
LLEN.
 155 PRINT"■   BEENDEN SIE IHRE EINGABE
JEWEILS MIT   DER  CLR-TASTE.■
 160 PRINT"   --------------------------
---------
```

Bild 1. Programmausdruck



```
 165 PRINT"■  WENN SIE BEGINNEN WOLLEN,
DRUECKEN
 170 PRINT"  SIE BITTE JETZT DIE LEERTAS
TE (SPACE).
 175 GETA$:IFA$=""THEN175
 180 IFA$=" "THEN200
 200 PRINT"■":CLR
 210 GETA$:IFA$=""THEN210
 220 IFA$="■"THENB$=B$+"■":GOTO300
 230 B=ASC(A$):IFB=13ORB=17THENCLR:GOTO1
40
 240 IFB=145ORB=157THEN200
 250 PRINTA$;:B$=B$+A$:GOTO210
 300 PRINT"■":L=LEN(B$)
 310 FORN=1TOL:C$=MID$(B$,N,1):PRINTC$;
 320 IFC$="■"THENZ=Z-1
 330 IFC$="■"THENZ=Z-1
 340 IFASC(C$)=20THENZ=Z-2
 350 Z=Z+1:IFZ<960THENNEXT:GOTO310
 360 GETA$:IFA$=""THEN360
 370 IFASC(A$)=13THENCLR:GOTO200
```

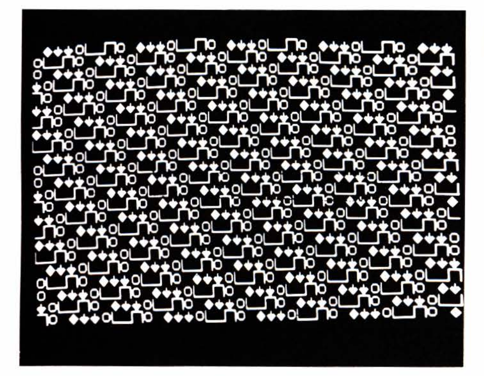
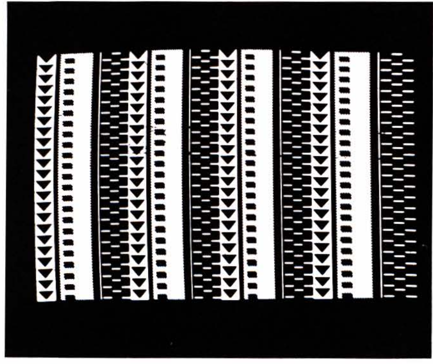
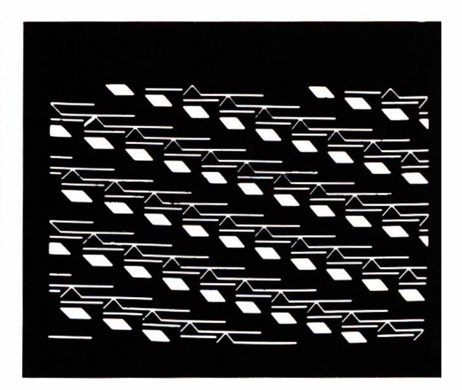




Bild 2. Solche und ähnliche Muster entwirft der PET mit Hilfe des Programmes von Bild 1

verse auf Normalbetrieb" hinzu und übergibt B-String an das Programmsegment „Grafik ausdrucken" in Zeile 300.

Andernfalls wird das eingegebene Zeichen darauf geprüft, ob es etwa einen Befehl mit den ASCII-Codes 13 oder 17 darstellt; das wären unzulässige Eingaben, die mit einem Rücksprung zur Darstellung der Bedienungsanleitung beantwortet werden (nach Zeile 140).

Schließlich untersucht der Rechner, ob er den Tastendruck als Korrekturverlangen zu werten hat. Ist das nicht der Fall, wird das eingegebene Zeichen dem Grundmuster in B-String hinzugefügt, und der Rechner wartet auf eine neue Eingabe.

Das letzte Programmsegment „Grafik ausdrucken" (ab Zeile 300) handhabt die gewünschten Wechsel zwischen Positiv- und Negativ-Darstellung, zählt ab, ob der Bildschirm schon gefüllt ist, und holt sich Element für Element aus B-String, um sie über den Ein-Zeichen-String C beim Ausdrucken lückenlos aneinanderzureihen. Schließlich wartet der Rechner in Zeile 360 ein Kommando ab und liest nach Drücken der Return-Taste eine neues Grundelement ein, wenn eine weitere Grafik produziert werden soll. Ist das nicht der Fall, meldet sich nach Betätigung der STOP-Taste, die hier nicht länger stört, das Betriebssystem wieder.

Bei der Eingabe der Grafik-Elemente hat man es in der Hand, ob das Endprodukt rechtwinkelig strukturiert ist. Das ist immer dann der Fall, wenn die Anzahl der druckenden Zeichen (Umschaltung von oder nach Negativ-Darstellung zählt nicht) einem gemeinsamen ganzzahligen Teiler mit 40, der Zahl der Plätze pro Zeile auf dem PET-Schirm, aufweist. Andernfalls ergeben sich diagonale Strukturen, wie sie einige Schirmfotos (*Bild 2*) zeigen.

Nach diesen notwendigerweise verhältnismäßig abstrakten und trockenen Erläuterungen nun einige Worte zu den praktischen Erfahrungen mit dem Programm „Grafik-Generator". Der PET verleitet, mit diesem Programm „im Bauch", immer wieder erneut zum Spielen. Es ist faszinierend, zu erleben, wie sich aus wenigen eingegebenen Zeichen auf überraschende Weise immer neue Muster von ästhetisch reizvollen Bildern ergeben. Über die zweckfreie Anwendung hinaus ist, so vermutet der Autor, ein Programm dieser Art auch für Leute von Wert, die Dinge wie Stoffmuster oder Tapeten zu entwerfen haben.

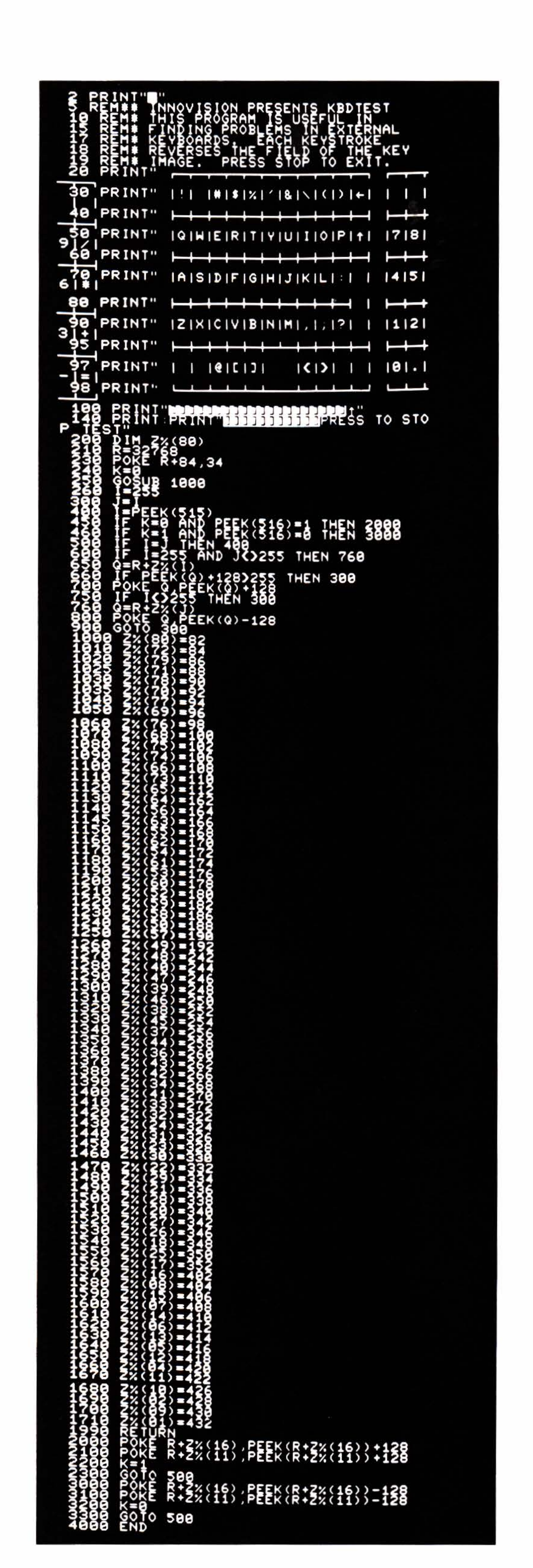
Hans-Georg Joepgen

„Tastentest" verrät unseriöse Kontakte:

Ein trickreiches Diagnose-Programm

Die Kundendienst-Werkstätten der Computer-Hersteller verfügen in der Regel über große Programmpakete, die den Technikern beim Warten und Reparieren von Rechnern die Arbeit erleichtern, sogenannte „Diagnose-Programme". Gewöhnlich werden diese Reperaturhilfen, deren Erstellung mit hohen Kosten verbunden ist, dem Kunden nicht zur Verfügung gestellt und auch vor der Konkurrenz geheimgehalten. Zwar laufen Diagnoseprogramme meist nur auf Modellen einer einzigen Computer-Familie, aber in ihnen verbirgt sich oft soviel an Ideenreichtum, daß man dem Mitbewerber auch nicht das Umschreiben für seine Rechner ermöglichen möchte. Verständlich – und um so dankbarer ist der Autor der Firma Commodore, die ihm dennoch die Veröffentlichung eines solchen firmeneigenen Diagnoseprogramms ermöglichte.

Das Programm „Tastentest" (in der englischen Originalfassung heißt es „Innovision", „Blick ins Innere") zeichnet ein Abbild der beiden Tastfelder des PET auf den Bildschirm. Beim Betätigen beliebiger Tasten, ausgenommen ist hier allein der RUN/STOP-Knopf, der zur Programmbeendigung dient, beim Drücken einer beliebigen Taste wird das zugeordnete Symbol auf dem Schirm invertiert dargestellt. In *Bild 1* hatte der Autor gerade den Zeigefinger auf der x-Taste zwi-



```
2 PRINT"█"
5 REM** INNOVISION PRESENTS KBDTEST
10 REM* THIS PROGRAM IS USEFUL IN
15 REM* FINDING PROBLEMS IN EXTERNAL
17 REM* KEYBOARDS.  EACH KEYSTROKE
18 REM* REVERSES THE FIELD OF THE KEY
19 REM* IMAGE.  PRESS STOP TO EXIT.
20 PRINT" ┌┬┬┬┬┬┬┬┬┬┬┬┐ ┌┬┬┐
30 PRINT" |!|  |#|$|%|'|&|\|(|)|←|  | | |
40 PRINT" ├┼┼┼┼┼┼┼┼┼┼┼┤ ├┼┼┤
50 PRINT" |Q|W|E|R|T|Y|U|I|O|P|↑|  |7|8|9|/|
60 PRINT" ├┼┼┼┼┼┼┼┼┼┼┼┤ ├┼┼┤
70 PRINT" |A|S|D|F|G|H|J|K|L|:|  |  |4|5|6|*|
80 PRINT" ├┼┼┼┼┼┼┼┼┼┼┤ | ├┼┼┤
90 PRINT" |Z|X|C|V|B|N|M|,|;|?|  |  |1|2|3|+|
95 PRINT" ├┼┼┼┼┼┼┼┼┼┼┼┤ ├┼┼┤
97 PRINT" | |  |@|[|]|   |<|>| | |  |0|.|-|=|
98 PRINT" └┴┴┴┴┴┴┴┴┴┴┴┘ └┴┴┘
100 PRINT"████████████████████↑"
140 PRINT:PRINT"██████████PRESS TO STO
P TEST"
200 DIM Z%(80)
210 R=32768
230 POKE R+84,34
240 K=0
250 GOSUB 1000
260 I=255
300 J=I
400 I=PEEK(515)
450 IF K=0 AND PEEK(516)=1 THEN 2000
460 IF K=1 AND PEEK(516)=0 THEN 3000
500 IF I=J THEN 400
600 IF I=255 AND J<>255 THEN 760
650 Q=R+Z%(I)
660 IF PEEK(Q)+128>255 THEN 300
700 POKE Q,PEEK(Q)+128
750 IF I<>255 THEN 300
760 Q=R+Z%(J)
800 POKE Q,PEEK(Q)-128
900 GOTO 300
1000 Z%(80)=82
1010 Z%(72)=84
1020 Z%(79)=86
1025 Z%(71)=88
1030 Z%(78)=90
1035 Z%(70)=92
1040 Z%(77)=94
1050 Z%(69)=96
1060 Z%(76)=98
1070 Z%(68)=100
1080 Z%(75)=102
1090 Z%(74)=106
1100 Z%(66)=108
1110 Z%(73)=110
1120 Z%(65)=112
1130 Z%(64)=162
1140 Z%(56)=164
1145 Z%(63)=166
1150 Z%(55)=168
1160 Z%(62)=170
1170 Z%(54)=172
1180 Z%(61)=174
1190 Z%(53)=176
1200 Z%(60)=178
1210 Z%(52)=180
1220 Z%(59)=182
1230 Z%(58)=186
1240 Z%(50)=188
1250 Z%(57)=190
1260 Z%(49)=192
1270 Z%(48)=242
1280 Z%(40)=244
1290 Z%(47)=246
1300 Z%(39)=248
1310 Z%(46)=250
1320 Z%(38)=252
1330 Z%(45)=254
1340 Z%(37)=256
1350 Z%(44)=258
1360 Z%(36)=260
1370 Z%(43)=262
1380 Z%(42)=266
1390 Z%(34)=268
1400 Z%(41)=270
1410 Z%(33)=272
1420 Z%(32)=322
1430 Z%(24)=324
1440 Z%(31)=326
1450 Z%(23)=328
1460 Z%(30)=330
1470 Z%(22)=332
1480 Z%(29)=334
1490 Z%(21)=336
1500 Z%(28)=338
1510 Z%(20)=340
1520 Z%(27)=342
1530 Z%(26)=346
1540 Z%(18)=348
1550 Z%(25)=350
1560 Z%(17)=352
1570 Z%(16)=402
1580 Z%(08)=404
1590 Z%(15)=406
1600 Z%(07)=408
1610 Z%(14)=410
1620 Z%(06)=412
1630 Z%(13)=414
1640 Z%(05)=416
1650 Z%(12)=418
1660 Z%(04)=420
1670 Z%(11)=422
1680 Z%(10)=426
1690 Z%(02)=428
1700 Z%(09)=430
1710 Z%(01)=432
1990 RETURN
2000 POKE R+Z%(16),PEEK(R+Z%(16))+128
2100 POKE R+Z%(11),PEEK(R+Z%(11))+128
2200 K=1
2300 GOTO 500
3000 POKE R+Z%(16),PEEK(R+Z%(16))-128
3100 POKE R+Z%(11),PEEK(R+Z%(11))-128
3200 K=0
3300 GOTO 500
4000 END
```

Bild 2. Blick in die „Geheimküche der Computer-Firmen: Mit diesem Programm verschaffen sich Wartungstechniker blitzartig einen Überblick über den Zustand der 72 Kontakte im Bedienfeld des PET

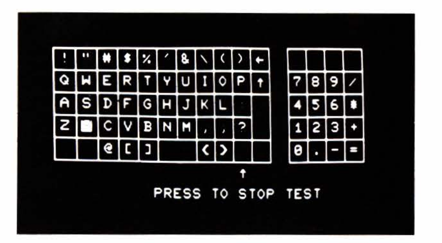


Bild 1 ► So anschaulich meldet das Programm „Tastentest" ob etwa Kontakte in den beiden Bedienfeldern des PET unzuverlässig geworden sind

schen z und c. Unterbleibt die Rückmeldung vom Schirm, liegt Unterbrechung vor; erscheint das zugeordnete Zeichen dauernd schwarz in weißem Feld, führt die Taste Dauerkurzschluß; bei Wackelkontakten flackert es – was will man mehr?

Ein Blick in die Programmliste zeigt, daß der PET hier unter Umgehung der üblichen Eingabeprozeduren „GET" und „INPUT", die für diesen Zweck ungeeignet wären, mit dem „Poke"-Befehl unmittelbar in den RAM-Bereich schreibt, der den Auffrisch-Speicher für den Schirm bildet; es sind dies die Dezimaladressen 32768...33767 für die 1000 Felder, aus denen der PET sein Schirmbild zusammensetzt. Die dort abgelegten Zeichen haben nur zum Teil ASCII-Format. Wichtig im vorliegenden Fall die Sonderzeichen dezimal 18 und dezimal 146. Sie bewirken Umschaltung der Umcodierung in Positiv-Darstellung, schwarzes Zeichen in weißem Feld, und Rückschaltung in die Normal-Betriebsart Negativ-Darstellung.

Das Programm zeigt in anschaulicher Weise, daß Diagnose-Routinen keine trockenen und abstrakten Rückmeldungen bewirken müssen, sondern daß sie ihre Ergebnisse durchaus auch in Form von Grafikdarstellungen mitteilen können, die man mit einem Blick erfaßt. Darüber hinaus kann das Programm als Anregung für PET-Hobbyprogrammierer dienen, bei Tastenabfragen und Schirmdarstellungen auch mal unorthodoxe Wege zu gehen.

Hans-Georg Joepgen

TRS-80 bekommt „Flügel"

Der preiswerte Heimcomputer TRS-80 von der Firma Radio Shack kann mit zahlreichen Zusätzen, die jetzt lieferbar sind, zum leistungsfähigen System ausgebaut werden. Eine wesentlich anspruchsvollere BASIC-Version, als im Grundgerät vorhanden, ist für 350 Fr. in ROMs erhältlich. Sie ist die Voraussetzung für den Anschluß eines Erweiterungs-Interface (1050 Fr.), das z. B. 16-K-RAM-Platinen oder die Steuereinheiten für Floppy-Disk-Laufwerke und einen neuen Zeilendrucker aufnehmen kann. Dieser Zeilendrucker wird für 4550 Fr. angeboten und kann in einer Sekunde 60...110 Zeichen zu Papier bringen. Eine preiswertere Druckerausführung stellt der sog. Screen Printer dar, der einfach den Inhalt des Bildschirms auf Knopfdruck wiedergibt. Eine passende Mini-Floppy-Disk-Station ist für 1750 Fr. zu haben. An Kassetten-Software wird neben einer großen Zahl von Spielen auch das sog. T-BUG-Programm (60 Fr.) angeboten, das den Zugriff zur Maschinensprache des Mikroprozessors Z-80 erlaubt. Außerdem kann der Benutzer mit Editor/Assembler arbeiten, wenn er sich ein Programm für 120 Fr. anschafft und 16 K RAM zur Verfügung hat.

□ Vertrieb: Comicro AG, Badenerstr. 281, CH-8003 Zürich, Tel. (01) 2 42 26 03 (in Deutschland: Tandy).

ELIZA – oder der Computer als Psychoanalytiker

Sich mit einem Computer zu unterhalten als wäre er ein menschliches Wesen, gehört sicher zu den reizvollsten und eindruckvollsten Vorführobjekten für den Hobbyisten. Wenn sich die Maschine noch dazu als einfallsreicher und schlagfertiger Gesprächspartner entpuppt, dann macht die Sache doppelt Spaß. ELIZA – so heißt das vorliegende Programm – stellt einen Psychoanalytiker dar, dem man seine Probleme anvertrauen kann. Leider versteht ELIZA nur englisch (wegen des einfacheren Aufbaus der englischen Sprache). Allerdings kann man beliebige Sätze eingeben: je länger um so besser! Das Programm wurde ursprünglich von Joseph Weizenbaum in der Programmiersprache LISP entwickelt. Diese Version basiert auf einer Beschreibung von Steve North in der Zeitschrift Creative Computing

Wirkungsweise

ELIZA nimmt einen Eingabesatz und analysiert ihn. Als erstes sucht sie nach Schlüsselworten, die in einer Tabelle auf Adresse 1000 zu finden sind. Als nächstes wird der Rest des Satzes von diesem Schlüsselwort an konjugiert, derart, daß z. B. aus dem Wort I das Wort YOU wird etc. Es steht Eliza eine Reihe von Antwortsätzen zur Verfügung, die anhand der Schlüsselworte über eine Tabelle auf Adresse 2530 ausgesucht werden. Bei den Antwortsätzen mit einem * am Ende des Satzes wird der konjugierte Antwortsatz angefügt. *Bild 1* zeigt die Programmliste. In *Bild 2* ist ein Bei-

```
1 REM            E L I Z A
2 REM Modifizierte Version aus Creative Computing
3 REM JUL/AUG 1977
4 REM
5 REM
6 REM BETRIEBSMITTELSTECKBRIEF
7 REM
8 REM COMPUTERTYP:                 Z80 S100 SYSTEM
9 REM PERIFERIE:                   DATENSICHTGERAET
10 REM SPEICHERBEDARF:             12K FUER BASIC U. 10K FUER ELIZA
11 REM BESONDERE BASIC BEFEHLE:
12 REM                             LINE INPUT,RESTORE n,INSTR
13 REM                             MID$,LEFT$,RIGHT$,LEN
14 REM GRENZEN DER ANWENDUNG:      -
15 REM BEMERKUNGEN:                12K ZAPPLE BASIC VON TDL
16 REM
 7 REM
90 CLEAR 500'STRING SPACE FESTLEGEN
100 DIM S(38),R(38),N(38)
110 N1=38:N2=12:N3=115
120 RESTORE 2530
130 FOR X=1 TO N1
140 READ S(X),L: R(X)=S(X): N(X)=S(X)+L-1
150 NEXT X
160 PRINT "HI, I AM ELIZA TELL ME YOUR PROBLEM"
170 LINE INPUT I$ 'EINGABE EINER ZEILE VOM BENUTZER
180 I$=" "+I$+"  "
190 REM BESEITIGEN VON APOSTROPHEN
200 FOR L=1 TO LEN(I$)
210 IF MID$(I$,L,1)="'" THEN I$=LEFT$(I$,L-1)+RIGHT$(I$,LEN(I$)-L):GOTO
210
220 NEXT L
230 IF I$=P$ THEN PRINT "PLEASE DONT REPEAT YOURSELF":GOTO 170
270 REM SUCHEN DES ERKENNUNGSWORTES
280 RESTORE
 70 S=0
300 FOR K=1 TO N1
305 IF S>0 THEN 340
310 READ K$
320 A=INSTR(I$,K$)
330 IF A<>0 THEN S=K:T=A:F$=K$
340 NEXT K
365 IF S>0 THEN K=S:L=T:GOTO 400
370 K=N1:GOTO 570
390 REM KONJUGATIONS PHASE
400 RESTORE 1200
410 C$=" "+RIGHT$(I$,LEN(I$)-LEN(F$)-L+1)
420 FOR X=1 TO N2/2
430 READ S$,R$
440 FOR L=1 TO LEN(C$)
450 IF L+LEN(S$)>LEN(C$) THEN 510
480 IF MID$(C$,L,LEN(S$))<>S$ THEN 510
490 C$=LEFT$(C$,L-1)+R$+RIGHT$(C$,LEN(C$)-L-LEN(S$)+1)
495 L=L+LEN(R$)
500 GOTO 540
510 IF L+LEN(R$)>LEN(C$) THEN 540
520 IF MID$(C$,L,LEN(R$))<>R$ THEN 540
530 C$=LEFT$(C$,L-1)+S$+RIGHT$(C$,LEN(C$)-L-LEN(R$)+1)
540 NEXT L
550 NEXT X
555 IF MID$(C$,2,1)=" "THEN C$=RIGHT$(C$,LEN(C$)-1)'EIN SPACE IN ANTWORT
560 REM ANTWORTSATZ FINDEN
570 RESTORE 1300
580 FOR X=1 TO R(K):READ F$:NEXT X' POSITIONIEREN
590 R(K)=R(K)+1:IF R(K)>N(K) THEN R(K)=S(K)
600 IF INSTR(F$,"*")=0 THEN PRINT F$:P$=I$:GOTO 170
610 A=INSTR(F$,"*")
620 PRINT LEFT$(F$,A-1);C$;RIGHT$(F$,LEN(F$)-A)
630 GOTO 170
1000 REM KEYWORDS
1050 DATA"CAN YOU","CAN I","YOU ARE","YOURE","I DONT","I FEEL"
1060 DATA "WHY DONT YOU","WHY CANT I","ARE YOU","I CANT","I AM","IM "
1070 DATA"YOU ","I WANT","WHAT","HOW","WHO","WHERE","WHEN","WHY"
1080 DATA"NAME","CAUSE","SORRY","DREAM","HELLO","HI ","MAYBE"
1090 DATA" NO","YOUR","ALWAYS","THINK","ALIKE","YES","FRIEND"
1100 DATA"COMPUTER","ELIZA","MONEY","NOKEYFOUND"
1200 REM KONJUGATIONEN
1230 DATA" ARE "," AM ","WERE ","WAS "," YOU "," I ","YOUR ","MY "
1235 DATA" IVE "," YOUVE "," IM "," YOURE "
1300 REM ANTWORTEN
1330 DATA"DON'T YOU BELIEVE THAT I CAN*"
1340 DATA"PERHAPS YOU WOULD LIKE TO BE ABLE TO*"
1350 DATA"YOU WANT ME TO BE ABLE TO*"
1360 DATA"PERHAPS YOU DON'T WANT TO*"
1365 DATA"DO YOU WANT TO BE ABLE TO*"
1370 DATA"WHAT MAKES YOU THINK I AM*"
1380 DATA "DOES IT PLEASE YOU TO BELIEVE I AM*"
1390 DATA"PERHAPS YOU WOULD LIKE TO BE*"
 00 DATA"DO YOU SOMETIMES WISH YOU WERE*"
1410 DATA"DON'T YOU REALLY*"
1420 DATA"WHY DON'T YOU*"
1430 DATA"DO YOU WISH TO BE ABLE TO*"
1440 DATA"DOES THAT TROUBLE YOU?"
1450 DATA"TELL ME MORE ABOUT SUCH FEELINGS."
1460 DATA"DO YOU OFTEN FEEL*"
1470 DATA"DO YOU ENJOY FEELING*"
1480 DATA"DO YOU REALLY BELIEVE I DON'T*"
1490 DATA"PERHAPS IN GOOD TIME I WILL*"
1500 DATA"DO YOU WANT ME TO*"
1510 DATA"DO YOU THINK YOU SHOULD BE ABLE TO*"
1520 DATA"WHY CAN'T YOU*"
1530 DATA"WHY ARE YOU INTERESTED IN WHETHER OR NOT I AM*"
1540 DATA"WOULD YOU PREFER IF I WHERE NOT*"
1550 DATA"PERHAPS IN YOUR FANTASIES I AM*"
1560 DATA"HOW DO YOU KNOW YOU CAN'T*"
1570 DATA"HAVE YOU TRIED?"
1580 DATA"PERHAPS YOU CAN NOW*"
 590 DATA"DID YOU COME TO ME BECAUSE YOU ARE*"
1600 DATA"HOW LONG HAVE YOU BEEN*"
1610 DATA"DO YOU BELIEVE IT IS NORMAL TO BE*"
1620 DATA"DO YOU ENJOY BEING*"
1630 DATA"WE WERE DISCUSSING YOU-- NOT ME."
1640 DATA"OH, I*"
1650 DATA"YOU'RE NOT REALLY TALKING ABOUT ME, ARE YOU?"
1660 DATA"WHAT WOULD IT MEAN TO YOU IF YOU GOT*"
1670 DATA"WHY DO YOU WANT*"
1680 DATA"SUPPOSE YOU SOON GOT*"
1690 DATA"WHAT IF YOU NEVER GOT*"
1700 DATA"I SOMETIMES ALSO WANT*"
1710 DATA"WHY DO YOU ASK?"
1720 DATA"DOES THAT QUESTION INTEREST YOU?"
1730 DATA"WHAT ANSWER WOULD PLEASE YOU THE MOST?"
1740 DATA"WHAT DO YOU THINK?"
1750 DATA"ARE SUCH QUESTIONS IN YOUR MIND OFTEN?"
1760 DATA"WHAT IS THAT YOY REALLY WANT TO KNOW?"
1770 DATA"HAVE YOU ASKED ANYONE ELSE?"
1780 DATA"HAVE YOU ASKED SUCH QUESTIONS BEFORE?"
1790 DATA"WHAT ELSE COMES TO MIND WHEN YOU ASK THAT?"
1800 DATA"NAMES DON'T INTEREST ME."
1810 DATA"I DON'T CARE ABOUT NAMES-- PLEASE GO ON."
1820 DATA"IS THAT THE REAL REASON?"
1830 DATA"DON'T ANY OTHER REASONS COME TO MIND?"
1840 DATA"DOES THAT REASON EXPLAIN ANYTHING ELSE?"
1850 DATA"WHAT OTHER REASONS MIGHT THERE BE?"
1860 DATA"PLEASE DON'T APOLOGIZE!"
1870 DATA"APOLOGIES ARE NOT NECESSARY."
1880 DATA"WHAT FEELINGS DO YOU HAVE WHEN YOU APOLOGIZE?"
1890 DATA"DON'T BE SO DEFENSIVE!"
1900 DATA"WHAT DOES THAT DREAM SUGGEST TO YOU?"
1910 DATA"DO YOU DREAM OFTEN?"
1920 DATA"WHAT PERSONS APPEAR IN YOUR DREAMS?"
1930 DATA"ARE YOU DISTURBED BY YUOR DREAMS?"
1940 DATA"HOW DO YOU DO ... PLEASE STATE YOUR PROBLEM."
1950 DATA"YOU DON'T SEEM QUITE CERTAIN."
1960 DATA"WHY THE UNCERTAIN TONE?"
1970 DATA"CAN'T YOU BE MORE POSITIVE?"
1980 DATA"YOU AREN'T SURE?"
1990 DATA"DON'T YOU KNOW?"
2000 DATA"WHY NO*"
2010 DATA"DONT SAY NO ITS ALWAYS SO NEGATIVE"
2020 DATA"WHY NOT?"
2030 DATA"ARE YOU SURE?"
2040 DATA"WHY NO?"
2050 DATA"WHY ARE YOU CONCERNED ABOUT MY*"
2060 DATA"WHAT ABOUT YOUR OWN*"
2070 DATA"CAN'T YOU THINK OF A SPECIFIC EXAMPLE?"
2080 DATA"WHEN?"
2090 DATA"WHAT ARE YOU THINKING OF?"
2100 DATA"REALLY, ALWAYS?"
2110 DATA"DO YOU REALLY THINK SO?"
  20 DATA"BUT YOU ARE NOT SURE YOU*"
```

```
2130 DATA"DO YOU DOUBT YOU*"
2140 DATA"IN WHAT WAY?"
2150 DATA"WHAT RESEMBLANCE DO YOU SEE?"
2160 DATA"WHAT DOES THE SIMILARITY SUGGEST TO YOU?"
2170 DATA"WHAT OTHER CONNECTIONS DO YOU SEE?"
2180 DATA"COULD THERE REALLY BE SOME CONNECTIONS?"
2190 DATA"HOW?"
2200 DATA"YOU SEEM QUITE POSITIVE."
2210 DATA"ARE YOU SURE?"
2220 DATA"I SEE."
2230 DATA"I UNDERSTAND."
2240 DATA "WHY DO YOU BRING UP THE TOPIC OF FRIENDS?"
2250 DATA"DO YOUR FRIENDS WORRY YOU?"
2260 DATA"DO YOUR FRIENDS PICK ON YOU?"
2270 DATA"ARE YOU SURE YOU HAVE ANY FRIENDS?"
2280 DATA"DO YOU IMPOSE ON YOUR FRIENDS?"
2290 DATA"PERHAPS YOUR LOVE FOR FRIENDS WORRIES YOU."
2300 DATA"DO COMPUTERS WORRY YOU?"
 310 DATA"ARE YOU TALKING ABOUT ME IN PARTICULAR?"
2320 DATA"ARE YOU FRIGHTENED BY MACHINES?"
2330 DATA"WHY DO YOU MENTION COMPUTERS?"
2340 DATA"WHAT DO YOU THINK MACHINES HAVE TO DO WITH YOUR PROBLEM?"
2350 DATA"DON'T YOU THINK COMPUTERS CAN HELP PEOPLE?"
2360 DATA"WHAT IS IT ABOUT MACHINES THAT WORRIES YOU?"
2370 DATA"SAY,DO YOU HAVE ANY PSYCHOLOGICAL PROBLEMS?"
2380 DATA"WHAT DOES THAT SUGGEST TO YOU?"
2390 DATA"I SEE"
2400 DATA"I'M NOT SURE I UNDERSTAND YOU FULLY"
2410 DATA"COME COME ELUCIDATE YOUR THOUGHTS"
2420 DATA"CAN YOU ELABORATE ON THAT ?"
2430 DATA "THAT IS QUITE INTERESTING."
2440 DATA "WHY DO YOU HAVE PROBLEMS WITH MONEY?"
2450 DATA "DO YOU THINK MONEY IS EVERYTHING?"
2460 DATA "ARE YOU SURE THAT MONEY IS THE PROBLEM?"
2470 DATA "I THINK WE WANT TO TALK ABOUT YOU,NOT ABOUT ME"
2480 DATA "WHATS ABOUT ME?"
2490 DATA "WHY DO YOU ALWAYS BRING UP MY NAME?"
2530 DATA 1,3,4,2,6,4,6,4,10,4,14,3,17,3,20,2,22,3,25,3
2540 DATA 28,4,28,4,32,3,35,5,40,9,40,9,40,9,40,9,40,9,40,9
2550 DATA 49,2,51,4,55,4,59,4,63,1,63,1,64,5,69,5,74,2,76,4
2560 DATA 80,3,83,7,90,3,93,6,99,7,116,3,113,3,106,7
```

Bild 1. Programmausdruck

```
HI, I AM ELIZA TELL ME YOUR PROBLEM
I AM A PRINTER
DID YOU COME TO ME BECAUSE YOU ARE A PRINTER
YES, I WANT TO HAVE AN INTELLIGENT SUBSYSTEM
WHAT WOULD IT MEAN TO YOU IF YOU GOT TO HAVE AN INTELLIGENT SUBSYSTEM
I HOPE TO BECOME SMARTER
SAY,DO YOU HAVE ANY PSYCHOLOGICAL PROBLEMS?
MY FRIENDS THE LINE PRINTERS ARE MY PROBLEM
WHY DO YOU BRING UP THE TOPIC OF FRIENDS?
BECAUSE THEY ARE MY PROBLEM
IS THAT THE REAL REASON?
YES, BECAUSE I THINK SO
DON'T ANY OTHER REASONS COME TO MIND?
SORRY, BUT NO
PLEASE DON'T APOLOGIZE!
WHY AM I APOLOGIZING
WHY DO YOU ASK?
BECAUSE IT IS INTERESTING WHY YOU SAY THIS
WE WERE DISCUSSING YOU-- NOT ME.
BUT I THINK YOU ARE THE COMPUTER I NEED FOR THE SUBSYSTEM
WHAT MAKES YOU THINK I AM THE COMPUTER YOU NEED FOR THE SUBSYSTEM
YOUR SYNTAX ERRORS AND YOUR FRONTPANEL
WHY ARE YOU CONCERNED ABOUT MY SYNTAX ERRORS AND MY FRONTPANEL
MAYBE YOU ARE NOT ALWAYS SPEAKING PROPERLY
DOES IT PLEASE YOU TO BELIEVE I AM NOT ALWAYS SPEAKING PROPERLY
MAYBE
YOU DON'T SEEM QUITE CERTAIN.
BECAUSE I AS WELL MAKE SOME MISTAKES IN SPELLING WORDS
DOES THAT REASON EXPLAIN ANYTHING ELSE?
ANYONE MAKES ERRORS
WHAT DOES THAT SUGGEST TO YOU?
THE PROBLEM IS CLEAR, I NEED A COMPUTER
DO COMPUTERS WORRY YOU?
SOMETIMES THEY DO
I SEE
```

Bild 2. Beispiel einer „Sitzung" mit ELIZA

spielprogramm aufgelistet, das auch verwendet werden kann, um zu testen, ob das Programm richtig eingegeben wurde. Es ist relativ leicht, den Wortschatz von ELIZA zu erweitern und eigene Antwortsätze hinzuzufügen. Dazu können die neuen Worte bzw. Antwortsätze in die entsprechenden Tabellen eingetragen werden. In Zeile 100 und 110 müssen die neuen Werte der Anzahl von Worten bzw. Sätzen eingetragen werden. N1 ist die Anzahl von Schlüsselworten, N2 die Anzahl der Konjugationen und N3 die Anzahl der Antwortsätze. In die Datentabelle bei 2530 muß ebenfalls eine Eintragung durchgeführt werden. Es wird dazu am Schluß der Tabelle die Zeilennummer (relativ) der neuen Antwortsätze eingefügt und dann die Zahl der neuen Antwortsätze für das betreffende Schlüsselwort.

Rolf Dieter Klein

Literatur

1 ELIZA. Creative Computing, Juli/Aug. 1977, S. 100 ff.

Computer stellen sich selbst ein Bein:

„Kommissar" deckt Rechenungenauigkeit auf

Unter Kriminalisten gilt jener Kollege als besonders erfolgreich, der einen lügenden Täter oder unzuverlässigen Zeugen allein durch geschickte Fragestellung, ohne weitere Beweismittel, in Widersprüche verwickelt und damit überführt. Nun, auch Computer lügen bisweilen. Dann nämlich, wenn sie Zahlen mit sehr vielen gültigen Stellen mathematisch miteinander verknüpfen sollen, dabei die Formatgrenzen überschritten werden und der Rechner sich mit Aufrunden oder gar mit dem Weglassen der letzten Nachkommastellen behilft. Wann und bei welchen Operationen dies geschieht, ist von Rechnertyp zu Rechnertyp verschieden. Das folgende Programm „Kommissar", geschrieben für den PET, läuft auf nahezu jeder BASIC-Maschine und zwingt den Computer, seine Unzulänglichkeiten offen einzugestehen: in der Manier eines erfolgreichen Kriminalisten gewissermaßen.

Wie geschieht das? Betrachten wir einmal folgenden Klammerausdruck:

$$x = \left(1 + \frac{1}{a}\right)^b$$

Dabei sind a und b positive Zahlen. Man erkennt, daß der Wert für x um so größer wird, je größer man b macht, und man erkennt auch, daß ein Wachsen von a den umgekehrten Effekt hat: x wird kleiner.

Was aber passiert nun, wenn wir a und b im gleichen Maße wachsen lassen? Wir schreiben a = n und b = n:

$$x = \left(1 + \frac{1}{n}\right)^n$$

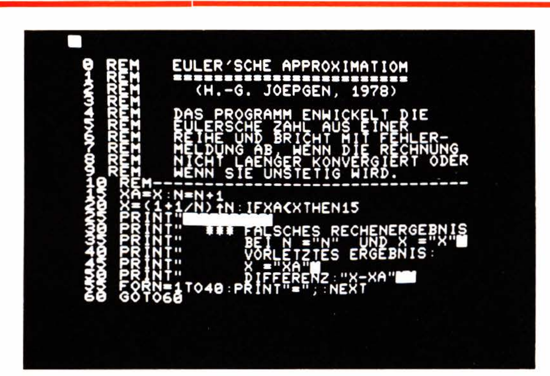


Bild 1. Mit Hilfe einer Näherungsrechnung, die *Leonhard Euler*, ein berühmter Mathematiker des 18. Jahrhunderts entwickelte, zwingt das Programm „Kommissar" den Computer, seine Schwächen einzugestehen. In diesem Programm bedeutet die Variable XA „X-Alt" und bezeichnet den x-Wert für ein um 1 kleineres n



Bild 2. Hier bekennt der PET, daß er bereits bei der 886. Rechnung passen muß. Programmierbare Taschenrechner werden übrigens in der Regel erst sehr viel später ungenau, sie verfügen über mehr „gültige Stellen" als die meisten BASIC-Maschinen

und erhalten beispielsweise folgende Wertetabelle:

n	x
1	2,000000
10	2,593742...
100	2,704813...
1000	2,7169239...
10000	2,7181459...
100000	2,7182682...
1000000	2,7182804...
10000000	2,7182818...

Zweierlei wird erkennbar: Eine Vergrößerung von n bedeutet in jedem Fall auch eine Erhöhung des Wertes von x. Zum zweiten aber sieht man, daß dieser Effekt um so kleiner wird, je größer n bereits ist. Irgendwann macht sich die weitere Vergrößerung von n nur noch in Nachkommastellen bemerkbar, die jenseits der vom Rechner noch beherrschbaren Stellenanzahl liegen.

Der „wahre Wert" von x für ein unendlich großes n ist übrigens die sogenannte „Eulersche Zahl" e, die Basis der natürlichen Logarithmen, eine wichtige Naturkonstante. Der Mathematiker *Leonhard Euler* (1707 bis 1783) hat sie mit Hilfe einer Annäherung, einer „Approximation", auf einige Dutzend Stellen genau berechnet.

Genau das, die Nachvollziehung der Fleißarbeit von Euler, gibt das Programm „Kommissar" dem Rechner als Aufgabe, und führt anschließend eine sogenannte Plausibilitätskontrolle durch. Es fragt, ob das vorletzte x auch wirklich kleiner war als das letzte x, das aufgrund eines um 1 höheren n errechnet wurde. Trifft das nicht mehr zu, dann meldet sich der Rechner und muß bekennen, daß er gelogen hat, weil er bei der Bereitstellung des Kehrwertes von n oder beim Potenzieren des Klammerausdrucks mit n nicht mehr mit der Anzahl der Stellen zurechtkam, mit denen er umgehen kann. *Hans-Georg Joepgen*

PET macht Musik

Für verschiedene Heimcomputer wird in letzter Zeit gebrauchsfertige Software auf Kassetten oder Floppy Disks angeboten, die sich selbst erklärt. D. h., der Computer fragt nach dem Einlesen des Programms den Benutzer: „Wollen Sie die Regeln wissen?", man braucht dann nur mit „ja" zu antworten, und die Anleitung zur Bedienung sowie die Programmdokumentation erscheinen auf dem Bildschirm. Die Amerikaner nennen das „Canned Software" (*can* = Dose). Ein Beispiel dafür ist ein Musikprogramm für den PET, das für 49 DM verkauft wird.

Zum Betrieb des Programms MUSIC wird ein 4...8-Ω-Lautsprecher über ein TTL-Gatter an den Port PA 0 (Stift C des parallelen Benutzer-Ports) gelegt.

Drei verschiedene Spielarten sind möglich:

1. Die Noten werden in einer Notenzeile eingegeben. Ein Pfeil zeigt auf die gerade zu spielende Note. Zur gleichen Zeit kann nur eine Note gespielt werden, die immer die gleiche Länge hat.

2. Die Noten werden als Zahlen codiert eingegeben. Die Dauer ist variabel.

3. Die Noten werden vom Gerät per Zufallsgenerator erzeugt. Die Geschwindigkeit richtet sich nach dem zuletzt eingegebenen numerischen Wert.

Betriebsart 1

RUN 200 eingeben, um den Bildschirm zu löschen, und Notenzeile, bestehend aus fünf Einzelzeilen,

„einzeichnen". Noten mit der Taste „Shift W" eingeben. Es sind die Noten D...G möglich. Benutzt wird zur Eingabe die Cursor-Steuerung des PET. Danach wird der Cursor auf eine freie Stelle gebracht und das Programm mit RUN gestartet. Nach kurzer Zeit, die zum Laden der Musikroutinen in Assemblersprache benötigt wird, beginnt der PET zu spielen, bis die STOP-Taste gedrückt wird.

Betriebsart 2

Nach dem Eingeben von RUN 600 antwortet der PET mit „ENTER MUSIC STRING?". Folgende Eingaben können nun gemacht werden:

- eine Zahl von 0 bis 9, die als Note interpretiert wird; daran anschließend eine weitere Zahl, die erste Zahl wird dann als Viertelnote aufgefaßt;
- eine Zahl, gefolgt von einen Punkt, wird als halbe Note interpretiert;
- eine Zahl, gefolgt von Buchstaben O, wird als ganze Note interpretiert.

Bis zu zwei Zeilen können eingegeben werden. Der Cursor kann zum Verbessern genutzt werden.

Betriebsart 3

Nach RUN 400 spielt der PET Noten in zufälliger Reihenfolge und anschließend die Tonleiter auf und ab. Eine niedrige Zahl von der Tastatur erhöht das Tempo, eine hohe verlangsamt es.

Kleines „Universalprogramm" macht aus SDK-85 Transientenrecorder, Logikanalysator und Datenempfangsstation

Der Mikrocomputer SDK-85, ausgelegt für Tastatur- und Konsolenbetrieb, verfügt bedauerlicherweise nicht über die Möglichkeit zum Einlesen von Programmen per Tonbandkassette. Wenn, wie beim Autor, keine Teletype-Maschine mit Lochstreifenleser zur Verfügung steht, muß man von Hand laden – das heißt, nach dem Einschalten jeden Befehl einzeln eintippen. Dies ist ein mühsames Geschäft. Da dem Verfasser ein Siemens-Hobbycomputer „Mikroset" mit Kassetten-Anschlußschaltung zur Verfügung steht, vertraute er seine Programme für den SDK-85 dem Mikroset an, programmierte ihn als „Programmsender" und entwarf ein kleines Miniprogramm für den Intel-Rechner, das ihn in die Lage versetzte, die vom Mikroset übertragenen Tonbanddaten in seine Speicher zu laden. Bald darauf stellte sich heraus, daß dieses Empfangsprogramm auch vorzüglich geeignet war, den SDK-85 ein wenig Logikanalysator und sogar Transientenrecorder spielen zu lassen – letzteres nach Änderung eines einzigen Befehls. Dieses kleine Universalprogramm und seine Anwendung sollen nachfolgend beschrieben werden.

Der Mikroset übermittelt dem Intel-Rechner seine Daten über ein zehnadriges Kabel. Eine Ader ist die Masseverbindung, acht Adern übertragen das 8-bit-Datenwort D 0...D 7, die letzte Ader „DG" („Datenwort jetzt gültig") dient der Synchronisation. Jedesmal, wenn auf dieser Steuerleitung ein H-Signal erscheint, übernimmt der SDK-85 das auf D 0...D 7 stehende Wort in jene Speicherstelle, deren Adresse im Doppelregister HL steht. Geht das Signal „DG" zurück nach L, erhöht der SDK den Inhalt von HL um 1 und wartet, bis ein erneutes H auf der Leitung DG die Übernahme des neuen auf D 0...D 7 stehenden Bitmusters in die nächste Speicherstelle veranlaßt – und so fort.

Fürwahr kein kompliziertes Programm (*Bild 1*), und dennoch recht leistungsfähig. Denn mit seiner Hilfe kann man nicht nur innerhalb weniger Sekunden umfangreiche Programme in den SDK übertragen, sondern auch schnelle Signalwechsel auf einer 8-bit-Leitung, die sich im Millisekunden-Bereich abspielen, später in aller Ruhe analysieren. Es ist hierzu lediglich erforderlich, die Leitung DG mit einem externen Takt-

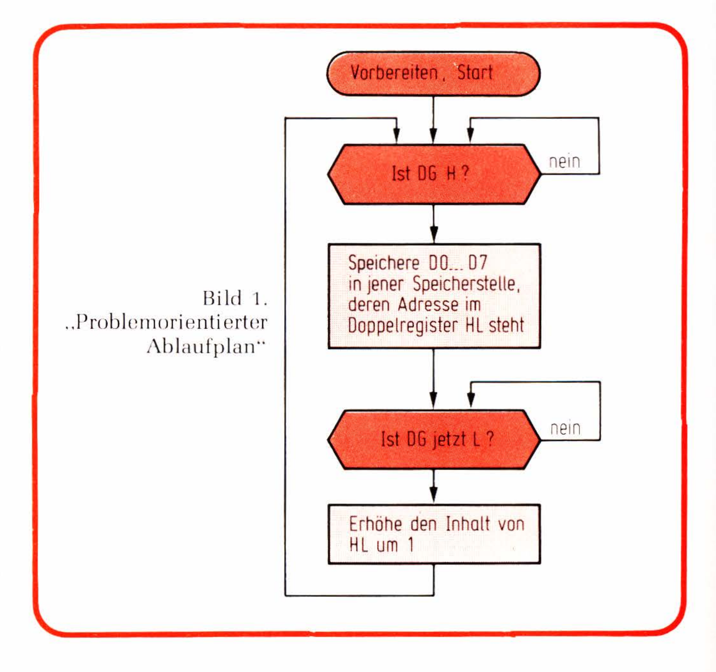


Bild 1. „Problemorientierter Ablaufplan"

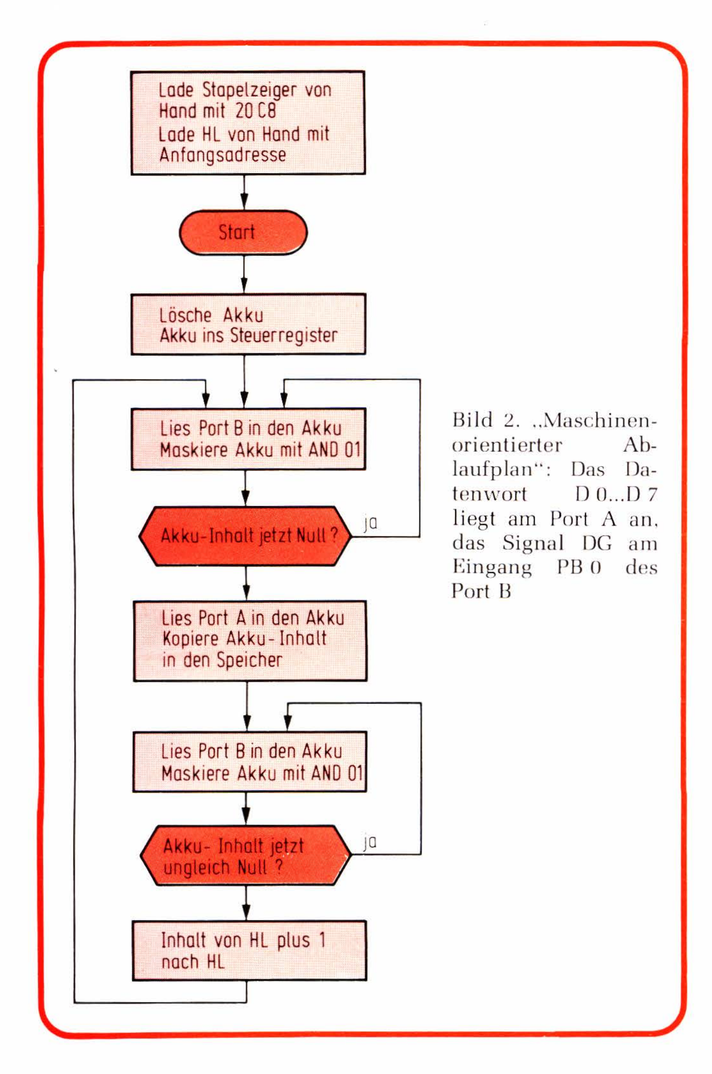


Bild 2. „Maschinenorientierter Ablaufplan": Das Datenwort D 0...D 7 liegt am Port A an, das Signal DG am Eingang PB 0 des Port B

Adresse für das Steuerregister der verwendeten Ein/Ausgangs-Ports: 28
Steuerwort für „Eingang ohne Quittung": 00
Adresse für Port A: 29 (D 0...D 7)
Adresse für Port B: 2A (PB0:DG)

Adresse	Befehl	Marke	Assembler-Code
2000	3E 00		MVI A, 00
2002	D3 28		OUT, 28
2004	DB 2A	P1	IN, 2A
2006	E6 01		ANI, 01
2008	CA 04 20		JZ, P1
200B	DB 29		IN, 29
200D	77		MOVM, A
200E	DB 2A	P2	IN, 2A
2010	E6 01		ANI, 01
2012	C2 0E 20		JNZ, P2
2015	23		INX H
2016	C3 04 20		JMP, P1

Bild 3. Dieses Mini-Programm verwandelt den SDK-85 in einen kleinen „Universal-Datenempfänger" und versetzt ihn in die Lage, sich von anderen Rechnern den Speicher volladen zu lassen. Stapelzeiger (*Stackpointer*) und Anfangsadresse in HL müssen vor dem Start von Hand geladen werden. Ersetzt man den Befehl in Speicherstelle 2015, Maschinencode 23, durch ein INR L (Maschinencode 2C), dann wird der SDK-85 zum Transientenrecorder, der die jeweils letzten 256 Zustände von D 0...D 7 speichert. Die Abtastrate bestimmt dabei ein externer Taktgenerator, der die Leitung PB 0 speist, das niedrigstwertige Bit des Ports B

generator anzusteuern. Pulst der Generator beispielsweise zwei Sekunden lang mit 100 Hz, so kann man nach dieser Prozedur in aller Ruhe untersuchen, welchen Zustand die acht Datenleitungen während dieser zwei Sekunden hatten, „gequantelt" in Abtastperioden, die jeweils 10 ms auseinanderliegen. Man muß dazu nur mit den Befehlen „Substitute Memory" und „Next" nachschauen, welche neuen Speicherinhalte der SDK-85 jetzt meldet. Eine elektronische Zeitlupe also, oder, wenn man so will, eine Art „dynamischer Logikanalysator".

Inkrementiert man nach jedem Zyklus nicht das Doppelregister HL, sondern allein das Register L, dann wird der SDK zum Transientenrecorder. Es finden sich dann zu jedem beliebigen Zeitpunkt im Speicher die Abbilder der jeweiligen Zustände auf den Leitungen D 0...D 7 in den letzten 256 Abtastperioden wieder, sobald das Programm durch elektrischen oder manuellen Restart-Befehl unterbrochen wird und der SDK zurück ins Betriebsprogramm springt.

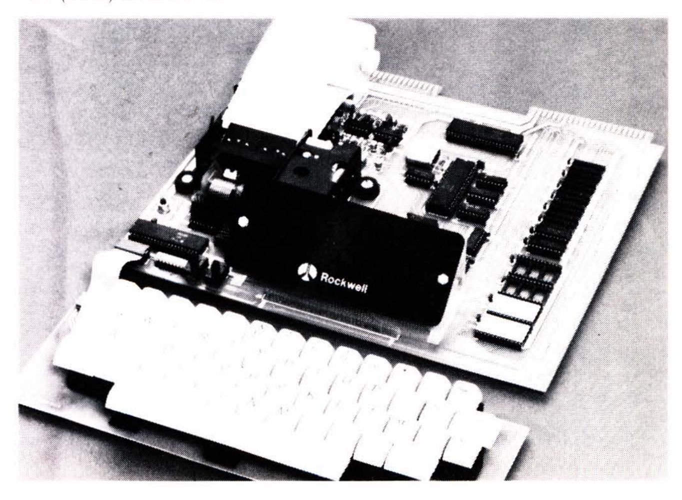
Diese Anwendung erfordert allerdings einen zusätzlichen RAM-Bereich auf der Intel-Karte, sonst schreibt der SDK ins Leere oder, schlimmer noch, überschreibt sein Programm mit den Daten, die er sammeln soll, interpretiert sie dann als Befehle und versucht sie zu befolgen. Das Ergebnis wäre ein totales Chaos.

Die Struktur des realisierten Programms ist aus *Bild 2* zu entnehmen, das Programm selbst ist in *Bild 3* aufgelistet. Viel Vergnügen mit diesem kleinen „Universalprogramm"! *Hans-Georg Joepgen*

KIM-Nachfolger ist da

Die Firma Rockwell stellt ein Miniatur-Entwicklungssystem vor, auf dem sämtliche Programme laufen, die auf dem bekannten KIM-1 erstellt wurden. Das Grundsystem wird für etwa 800 DM angeboten und enthält neben der eigentlichen Computerplatine mit Monitorprogramm, 1...4 K RAM, zwei 8-bit-Ports usw. eine alphanumerische Tastatur sowie einen Thermodrucker und eine 20stellige alphanumerische Leuchtanzeigeeinheit. Zusätzliche ROM-Sockel nehmen Bausteine des Typs 2332 oder 2716 auf. Der Benutzer kann darin Assembler, Texteditor oder BASIC-Interpreter unterbringen. Schnittstellen für 8-Kanal-Fernschreiber und zwei Kassettenlaufwerke sind ebenfalls vorhanden. Der Erweiterungsstecker ist mit dem des KIM-1 vollkommen identisch.

□ Vertrieb: Rockwell Int. GmbH, Fraunhoferstr. 11, 8033 Martinsried, Tel. (0 89) 8 59 95 75.



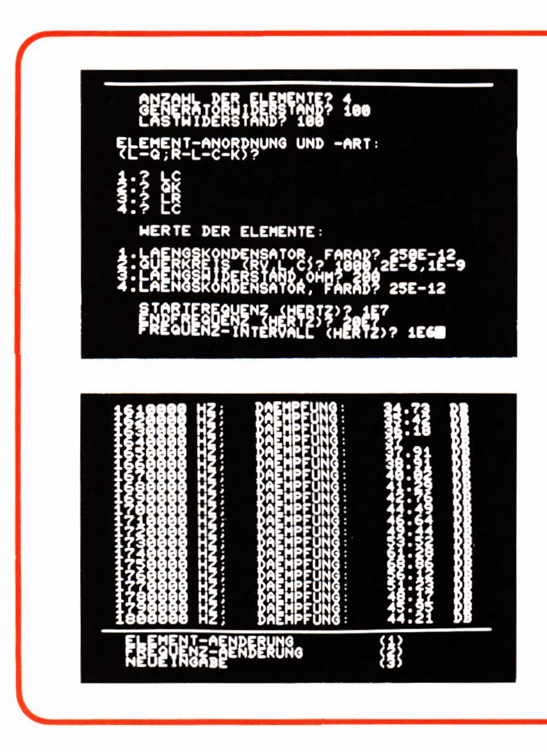
Hans-Georg Joepgen

„Wobbel-PET“ macht Vierpolrechnung schmackhaft: Simulation des Betriebsverhaltens passiver Netzwerke

Mit der Vierpolrechnung ist es eine vertrackte Sache. Auf der einen Seite erlaubt sie die elegante Vorausberechnung der Eigenschaften nahezu jeder beliebigen Schaltung, andererseits handelt es sich bei der Vierpol-Theorie um ein höchst abstraktes Gedankenwerk, das komplizierte Operationen mit imaginären Zahlen verlangt, um die Praktiker zumeist und Hobby-Elektroniker wohl fast immer am liebsten einen hohen Bogen schlagen. Doch das muß nicht sein: Wenn man die lästige und fehlerträchtige Kleinarbeit sowie die Berücksichtigung der hochwissenschaftlichen Theorie mit Hilfe des Programm-Paketes „Wobbel-PET“ dem Hobbycomputer PET 2001 oder einem anderen BASIC-Tischrechner überläßt, dann kann man die Vorzüge der Vierpoltheorie auch im Hobby-Labor oder am Arbeitsplatz nutzen, ohne ihre Nachteile – Fehleranfälligkeit beim Rechnen mit komplexen Zahlen sowie Unhandlichkeit – befürchten zu müssen.

Konkret: „Wobbel-PET“, angeregt durch eine Veröffentlichung des Vierpolspezialisten *Stefan Hamerli* [3] und unter seiner Beratung entstanden, ermöglicht die Eingabe beliebiger, unverzweigter passiver Netzwerke und die Untersuchung ihres stationären Betriebsverhaltens in Abhängigkeit von Frequenz, Abschlüssen und weiteren frei wählbaren Parametern. Je nach Anwendungsfall kann man sich aus den Segmenten des Programm-Paketes die Art der Untersuchung zusammenstellen, die der PET dann vornimmt. Im vorgeführten Fall wird beispielsweise die Betriebsdämpfung eines versteilerten Tiefpasses als Funktion der Frequenz tabellarisch ausgegeben.

Bereits hier zeigt sich die hohe Flexibilität von „Wobbel-PET“: Man kann nämlich eine Fülle von Parametern ändern, bis die Wunschlösung realisiert ist. Für die Praxis heißt das: Kein endloses Herumexperimentieren mit Wobbler, Sichtgerät und Lötkolben, bis die Filterkurve sitzt. Befürchtungen, theoretischer Verlauf auf dem PET-Schirm und tatsächliches Betriebsverhalten einer Schaltung könnten allzusehr auseinanderlaufen, sind unbegründet. Erstens berücksichtigt der Rechner bereits, daß es auf dieser Welt keine verlustfreien Spulen und Kondensatoren gibt, zweitens ist er darauf vorbereitet, daß die Generatorwiderstände der meisten handelsüblichen Wobbler eben nicht rein ohmisch sind, wie das die Datenblätter behaupten. Diskrepanzen zwischen errechneten und gemessenen Werten sind oft darauf zurückzuführen, daß sich Generatoren und Verbraucher höchst heimtückisch wider Erwarten als „komplexe Brüder“ be-



◄ Bild 1. So fragt der PET den Mann am Rechner nach den Ausgangsdaten.

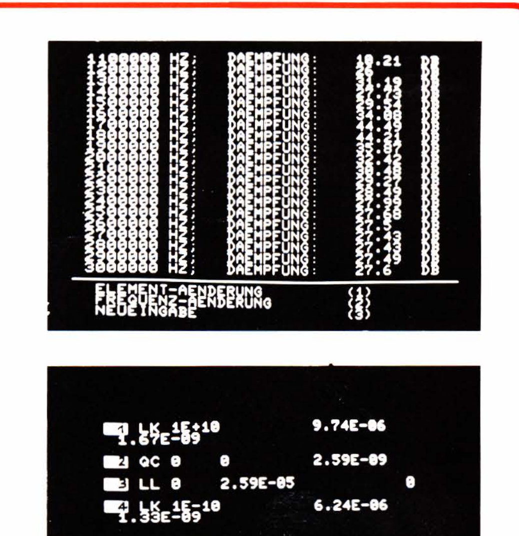


Bild 2. ► Test-Tiefpaß auf dem „Prüfstand“

Bild 3 (links unten). „Lupen-Effekt“: Ein besonders interessanter Bereich in der Nachbarschaft eines Dämpfungspoles wird erneut, diesmal mit höherer Auflösung, untersucht

Bild 4. ► Änderungen von Parametern, wenn die Dämpfungskurve nicht das ist, was der Vierpol-Entwerfer gern hätte: Kein Problem für den „Wobbel-PET“

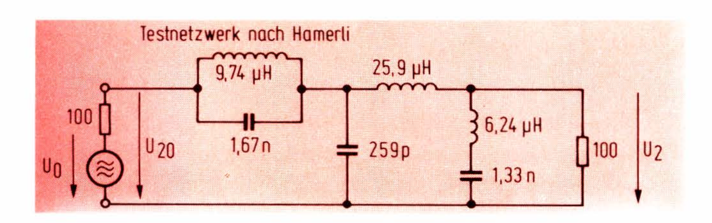


Bild 5. Dieses Testnetzwerk, ein versteilerter Tiefpaß nach Hamerli [3], muß für die Frequenz 1,4 MHz einen Dämpfungsbetrag von (teilgerundet) 27.7326 dB ergeben. Das Beispiel zeigt, wie ein Netzwerk zur Eingabe in seine Elemente zerlegt wird. Die Eingabereihenfolge lautet „LK, QC, LL, QK". Damit weiß der Rechner, daß es sich um die Kettenschaltung aus Längskreis, Querkondensator, Längsspule und Querkreis handelt. Offene Längskreise und geschlossene Querkreise werden durch die isolierte Eingabe ihrer Bauelemente dargestellt

nehmen. „Wobbel-PET" erlaubt die Eingabe und Berücksichtigung imaginärer Komponenten bei Netzwerk-Quelle und Netzwerk-Senke. – Also bitte!

Ein bißchen Theorie – aber nur ganz wenig

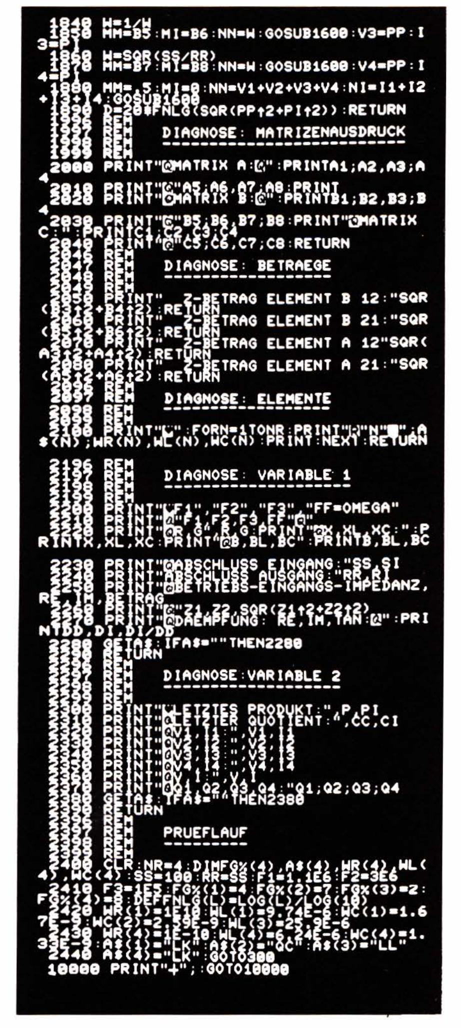
Zum Verständnis von Wobbel-PET muß man nur wissen, daß sich das Verhalten eines „Vierpols" – ein solches Element kann ein simples Bauteil sein, aber auch eine riesige Anordnung von Bauteilen – unter gewissen Einschränkungen durch acht Zahlen beschreiben läßt, die man aus Gründen der Handlichkeit zu jeweils vier Paaren (Matrixelemente) zusammenfaßt. Ein solches Zahlenpaar kann beispielsweise aus der ohmschen und der kapazitiven Komponente einer Serienschaltung von Kondensator und Widerstand bestehen.

Die vier Wertepaare also bilden nun eine „Matrix". Aus dieser Matrix kann man nahezu alle interessierenden Eigenschaften berechnen. Und nun kommt das Praktische an der Vierpolrechnung: Hat man eine Anzahl Bauteile und die Frequenz, um die es geht, kann man die Matrizen der Bauteile schreiben. Diese Matrizen muß man nur noch mathematisch miteinander verbinden, wobei die Art der Verbindung von der Anordnung der Bauteile abhängt, längs oder quer zum Beispiel. Damit bekommt man eine „Schluß-Matrix", wieder vier Elemente zu je zwei Werten, und an der kann man vielerlei Untersuchungen vornehmen, sie beschreibt für den Stationär-Fall, also mit Ausnahme von Einschwingvorgängen und auch mit Ausnahme nichtlinearer Effekte, das Verhalten einer ganzen Schaltung erschöpfend.

Das soll hier reichen, wer Genaueres wissen will, sei auf die angegebenen Veröffentlichungen und auf Lehrbücher der höheren Mathematik verwiesen.

```
11 REM  PASSIVE KETTEN-NETZWERKE
12 REM  ------------------------
14 REM  (COPYRIGTHT JOEPGEN,1978)
15 REM
16 CLR:DEFFNLG(L)=LOG(L)/LOG(10)
19 REM
20 REM  EINGABE 1
30 REM  ---------
40 PRINT"":FORN=1TO40:PRINT"-";:NEXT
50 INPUT"   ANZAHL DER ELEMENTE";NR
60 INPUT"   GENERATORWIDERSTAND";SS
70 INPUT"   LASTWIDERSTAND";RR:N=NR
80 DIMFG%(N),A$(N),WR(N),WL(N),WC(N)
90 PRINT" ELEMENT-ANORDNUNG UND -ART:
100 PRINT" (L-Q;R-L-C-K)?"
110 FORN=1TONR
120 PRINTN"].";:INPUTA$(N)
130 IFLEN(A$(N))<>2THEN120
135 IFLEFT$(A$(N),1)="L"THEN150
140 IFLEFT$(A$(N),1)="Q"THEN150
145 GOTO120
150 IFRIGHT$(A$(N),1)="R"THEN180
155 IFRIGHT$(A$(N),1)="L"THEN180
160 IFRIGHT$(A$(N),1)="C"THEN180
165 IFRIGHT$(A$(N),1)="K"THEN180
170 GOTO120
180 NEXT
200 PRINT"   WERTE DER ELEMENTE:"
210 FORN=1TONR:PRINTN"].";
215 IFA$(N)="LR"THENINPUT"LAENGSWIDERST
AND, OHM";WR(N):FG%(N)=1
220 IFA$(N)="LL"THENINPUT"LAENGSSPULE,
HENRY";WL(N):FG%(N)=2
225 IFA$(N)="LC"THENINPUT"LAENGSKONDENS
ATOR, FARAD";WC(N):FG%(N)=3
230 IFA$(N)="LK"THENINPUT"LAENGSKREIS (
RP,L,C)";WR(N),WL(N),WC(N):FG%(N)=4
235 IFA$(N)="QR"THEN INPUT"QUERWIDERSTA
ND, OHM";WR(N):FG%(N)=5
240 IFA$(N)="QL"THENINPUT"QUERSPULE, HE
NRY";WL(N):FG%(N)=6
245 IFA$(N)="QC"THENINPUT"QUERKONDENSAT
OR, FARAD";WC(N):FG%(N)=7
250 IFA$(N)="QK"THENINPUT"QUERKREIS (RV
,L,C)";WR(N),WL(N),WC(N):FG%(N)=8
255 NEXT
257 REM
258 REM  EINGABE 2
259 REM  ---------
260 INPUT"   STARTFREQUENZ (HERTZ)";F4
:IFF4<>0THENF1=F4
270 INPUT"   ENDFREQUENZ (HERTZ)";F5:IF
F5<>0THENF2=F5
280 INPUT"   FREQUENZ-INTERVALL (HERTZ)
";F6:PRINT:IFF6<>0THENF3=F6
297 REM
298 REM  HAUPTSCHLEIFE
299 REM  -------------
300 FF=2*π*F1:PRINT""
310 C1=1:C3=0:C5=0:C7=1
320 C2=0:C4=0:C6=0:C8=0
330 FORN=1TONR:GOSUB1100:REM TRANSFER
340 A1=1:A3=0:A5=0:A7=1
350 A2=0:A4=0:A6=0:A8=0
360 ONFG%(N)GOSUB1400,1425,1450,1475,15
00,1525,1550,1575
370 GOSUB1000:NEXT:GOSUB1100
380 GOSUB1800
390 DX=INT(D*100+.5)/100:FQ=INT(FF/(2*π
)*10+.5)/10
400 PRINTFQ"HZ;   DAEMPFUNG:   "DX;:PRI
NTTAB(36)"DB"
410 FF=FF+2*π*F3
420 IFINT(FF/(2*π))=<F2THEN310
496 REM
497 REM  SCHLUSSSCHLEIFE
498 REM  ---------------
499 REM
500 FORN=1TO40:PRINT"-";:NEXT
510 PRINT"  ELEMENT-AENDERUNG        (
1)"
520 PRINT"  FREQUENZ-AENDERUNG       (
2)"
530 PRINT"  NEUEINGABE               (
3)"
540 GETA$:IFA$=""THEN540
550 A=VAL(A$):IFA=0THEN540
```

```
570 IFA>3THEN540
580 ONAGOSUB600,260,16
600 GOSUB2100:INPUT"NO. AENDERUNG";N
610 INPUT"NEUWERTE, R,L,C";WR(N),WL(N),
WC(N):GOTO300
996 REM
997 REM  MATRIZENMULTIPLIKATION
998 REM  ----------------------
999 REM
1000 C1=A1*B1-A2*B2+A3*B5-A4*B6
1010 C2=A1*B2+A2*B1+A3*B6+A4*B5
1020 C3=A1*B3-A2*B4+A3*B7-A4*B8
1030 C4=A1*B4+A2*B3+A3*B8+A4*B7
1040 C5=A5*B1-A6*B2+A7*B5-A8*B6
1050 C6=A5*B2+A6*B1+A7*B6+A8*B5
1060 C7=A5*B3-A6*B4+A7*B7-A8*B8
1070 C8=A5*B4+A6*B3+A7*B8+A8*B7
1080 RETURN
1096 REM
1097 REM  MATRIZENTRANSFER
1098 REM  ----------------
1099 REM
1100 B1=C1:B2=C2:B3=C3:B4=C4:B5=C5
1110 B6=C6:B7=C7:B8=C8:RETURN
1196 REM
1197 REM  KOMPLEXE DIVISION
1198 REM  -----------------
1199 REM
1200 CC=(ZZ*NN+ZI*NI)/(NN*NN+NI*NI):CI=
(ZI*NN-ZZ*NI)/(NN*NN+NI*NI):RETURN
1210 Z1=CC:Z2=CI:RETURN
1296 REM
1297 REM  EINGANGSIMPEDANZ
1298 REM  ----------------
1299 REM
1300 V1=B1*RR-B2*RI:I1=B1*RI+B2*RR
1310 V2=B5*RR-B6*RI:I2=B5*RI+B6*RR
1320 ZZ=B3+V1:ZI=B4+I1:NN=B7+V2
1330 NI=B8+I2:GOSUB1200
1340 Z1=CC:Z2=CI:RETURN
1396 REM
1397 REM  EINZELMATRIZEN
1398 REM  --------------
1399 REM
1400 A3=WR(N):RETURN
1425 A4=FF*WL(N):RETURN
1450 A4=-1/(FF*WC(N)):RETURN
1475 G=1/WR(N):BL=-1/(FF*WL(N)):BC=FF*W
C(N):B=BC+BL:DR=G↑2+B↑2
1480 A3=G/DR:A4=-B/DR:RETURN
1500 A5=1/WR(N):RETURN
1525 A6=-1/(FF*WL(N)):RETURN
1550 A6=FF*WC(N):RETURN
1575 R=WR(N):XL=FF*WL(N):XC=-1/(FF*WC(N
)):X=XL+XC:DR=R↑2+X↑2
1580 A5=R/DR:A6=-X/DR:RETURN
1596 REM
1597 REM  KOMPLEXE MULTIPLIKATION
1598 REM  -----------------------
1599 REM
1600 PP=MM*NN-MI*NI
1610 PI=MM*NI+MI*NN:RETURN
1646 REM
1647 REM  DAEMPFUNG 1
1648 REM  -----------
1649 REM
1650 ZZ=1:ZI=0:NN=RR:NI=RI:GOSUB1200:Q1
=CC:Q2=CI
1660 ZZ=SS:ZI=SI:NN=RR:NI=RI:GOSUB1200:
Q3=CC:Q4=CI
1670 MM=B1:MI=B2:NN=B3:NI=B4:GOSUB1600:
V1=PP:V2=PI
1680 MM=V1:MI=V2:NN=Q1:NI=Q2:GOSUB1600:
V3=PP:V4=PI
1690 MM=B5:MI=B6:NN=SS:NI=SI:GOSUB1600:
V5=PP:V6=PI
1700 MM=B7:MI=B8:NN=Q3:NI=Q4:GOSUB1600:
V7=PP:V8=PI
1710 DD=V3+V5+V7:DI=V4+V6+V8
1720 D=20*FNLG(SQR(DD↑2+DI↑2)):RETURN
1797 REM
1798 REM  DAEMPFUNG 2
1799 REM  -----------
1800 W=SQR(RR/SS)
1810 MM=B1:MI=B2:NN=1/W:NI=0:GOSUB1600:
V1=PP:I1=PI
1820 W=1/SQR(SS*RR)
1830 MM=B3:MI=B4:NN=W:GOSUB1600:V2=PP:I
2=PI
```



```
1840 W=1/W
1850 MM=B5:MI=B6:NN=W:GOSUB1600:V3=PP:I
3=PI
1860 W=SQR(SS/RR)
1870 MM=B7:MI=B8:NN=W:GOSUB1600:V4=PP:I
4=PI
1880 MM=.5:MI=0:NN=V1+V2+V3+V4:NI=I1+I2
+I3+I4:GOSUB1600
1890 D=20*FNLG(SQR(PP↑2+PI↑2)):RETURN
1996 REM
1997 REM  DIAGNOSE: MATRIZENAUSDRUCK
1998 REM  --------------------------
1999 REM
2000 PRINT"MATRIX A:":PRINTA1;A2,A3;A
4
2010 PRINT""A5;A6,A7;A8:PRINT
2020 PRINT"MATRIX B:":PRINTB1;B2,B3;B
4
2030 PRINT""B5;B6,B7;B8:PRINT"MATRIX
C:":PRINTC1;C2,C3;C4
2040 PRINT""C5;C6,C7;C8:RETURN
2046 REM
2047 REM  DIAGNOSE: BETRAEGE
2048 REM  ------------------
2049 REM
2050 PRINT"  Z-BETRAG ELEMENT B 12:"SQR
(B3↑2+B4↑2):RETURN
2060 PRINT"  Z-BETRAG ELEMENT B 21:"SQR
(B5↑2+B6↑2):RETURN
2070 PRINT"  Z-BETRAG ELEMENT A 12"SQR(
A3↑2+A4↑2):RETURN
2080 PRINT"  Z-BETRAG ELEMENT A 21:"SQR
(A5↑2+A6↑2):RETURN
2096 REM
2097 REM  DIAGNOSE: ELEMENTE
2098 REM  ------------------
2099 REM
2100 PRINT"":FORN=1TONR:PRINT""N"";A
$(N);WR(N),WL(N),WC(N):PRINT:NEXT:RETURN
2196 REM
2197 REM  DIAGNOSE: VARIABLE 1
2198 REM  --------------------
2199 REM
2200 PRINT"F1","F2","F3","FF=OMEGA"
2210 PRINT""F1,F2,F3,FF""
2220 PRINT"R,G",R,G:PRINT"X,XL,XC:":P
RINTX,XL,XC:PRINT"B,BL,BC":PRINTB,BL,BC
2230 PRINT"ABSCHLUSS EINGANG:"SS,SI
2240 PRINT"ABSCHLUSS AUSGANG:"RR,RI
2250 PRINT"BETRIEBS-EINGANGS-IMPEDANZ,
RE,IM,BETRAG
2260 PRINT""Z1,Z2,SQR(Z1↑2+Z2↑2)
2270 PRINT"DAEMPFUNG: RE,IM,TAN:":PRI
NTDD,DI,DI/DD
2280 GETA$:IFA$=""THEN2280
2290 RETURN
2296 REM
2297 REM  DIAGNOSE:VARIABLE 2
2298 REM  -------------------
2299 REM
2300 PRINT"LETZTES PRODUKT:",P,PI
2310 PRINT"LETZTER QUOTIENT:",CC,CI
2320 PRINT"V1,I1:",V1,I1
2330 PRINT"V2,I2:",V2,I2
2340 PRINT"V3,I3:",V3,I3
2350 PRINT"V4,I4:",V4,I4
2360 PRINT"V,I:",V,I
2370 PRINT"Q1,Q2,Q3,Q4:"Q1;Q2;Q3;Q4
2380 GETA$:IFA$=""THEN2380
2390 RETURN
2396 REM
2397 REM  PRUEFLAUF
2398 REM  ---------
2399 REM
2400 CLR:NR=4:DIMFG%(4),A$(4),WR(4),WL(
4),WC(4):SS=100:RR=SS:F1=1.1E6:F2=3E6
2410 F3=1E5:FG%(1)=4:FG%(2)=7:FG%(3)=2:
FG%(4)=8:DEFFNLG(L)=LOG(L)/LOG(10)
2420 WR(1)=1E10:WL(1)=9.74E-6:WC(1)=1.6
7E-9:WC(2)=2.59E-9:WL(3)=25.9E-6
2430 WR(4)=1E-10:WL(4)=6.24E-6:WC(4)=1.
33E-9:A$(1)="LK":A$(2)="QC":A$(3)="LL"
2440 A$(4)="LK":GOTO300
10000 PRINT"+";:GOTO10000
```

Bild 6. Programmausdruck (in Zeile 1200 entfällt der RETURN-Befehl; in Zeile 2440 ist LK gegen QK zu ersetzen; Zeile 330: Anstelle von FOR N=1 TO NR erscheint FOR N=NR TO 1 STEP -1; Zeile 1810: NN=1/W wird ersetzt durch NN=W)

Erweiterung für Wobbel-PET

Das beschriebene Programm ist also fix und fertig zur Berechnung von Filterdämpfungen. Es bietet darüber hinaus alle elementaren Handreichungen, die man benötigt, um sich ein Sonderprogramm nach Maß zu schneidern. Anregungen dafür liegen in der Luft: Wenn man noch andere Zusammenschaltungs-Formen als die hier allein verwendete Kettenschaltung einführt, kann man mit beliebig verzweigten Vierpolen umgehen, durch Einführung aktiver Bauelemente gar ergibt sich eine schier unbegrenzte Anzahl von Variationsmöglichkeiten. Dazu allerdings sollte die angegebene Literatur zu Rate gezogen werden, wenn man noch nicht „vierpolfest" ist.

Eine Bemerkung zum Schluß: Der Autor ärgert sich oft über die unzureichende Dokumentation von BASIC-Programmen, die zwar wunderschön laufen, sich nahezu jeder Änderung aber entziehen, weil es einfach nicht gelingt, den Ablauf genügend transparent zu machen. Grund: unzureichender Begleittext. Hier hat der Autor einmal vorgeführt, wie er sich ein solches Programm dokumentiert wünscht: Und nun: Räumen Sie Ihren Wobbler zur Seite, wobbeln Sie mal *mit „Wobbel-PET"!*

Literatur

1 *Feldkeller, R.:* „Einführung in die Vierpoltheorie der elektrischen Nachrichtentechnik". Stuttgart 1976, S. Hirzel Verlag.

2 *Harm, H.:* Schaltungsanalyseprogramm für Kleinrechner. ELEKTRONIK 1975, H. 4, S. 101...106, H. 5, S. 83...86, H. 6, S. 78...82.

3 *Hamerli, S.:* Wobbelkurven ohne Meßgerät ermittelt. ELEKTRONIK 1978, H. 5, S. 50...53.

4 Röhre und Transistor als Vierpol. Herausgeber AEG-Telefunken, Fachbereich Röhren, Ulm.

Tabelle der Variablen

Hier werden, in der Reihenfolge ihres ersten Auftretens im Programm, die benutzten Variablen erläutert.

Name	Mnemotechnisches Bezugswort	Funktion
N	–	Schleifen-Zähler
NR	Nummer	Anzahl der Schaltungselemente
SS	–	Realteil der Generator-Impedanz
RR	–	Realteil der Lastimpedanz
FG%(N)	Flag	Schlüssel für Vielfachverzweigung*)
A$(N)	–	Schlüssel für Bauteil-Anordnung und -Art im Netzwerk*)
WR(N)	Wert für R	Ohmsche Komponente des Bauteils N
WL(N)	Wert für L	Induktive Komponente des Bauteils N
WC(N)	Wert für C	Kapazitive Komponente des Bauteils N
F4	Frequenz 4	Ersatzwert für F1
F5	Frequenz 5	Ersatzwert für F2
F6	Frequenz 6	Ersatzwert für F3
F1	Frequenz 1	Startfrequenz der Untersuchung
F2	Frequenz 2	Endfrequenz der Untersuchung
F3	Frequenz 3	Frequenz-Intervall
FF	–	Kreisfrequenz Omega
C1,C3, C5,C7	–	Realanteil der Koeffizienten einer 2x2-Matrix namens ‚C'
C2,C4, C6,C7	–	Imaginäranteil der Koeffizienten der Matrix C
A1...A8	–	Wie oben, für eine Matrix ‚A'
B1...B8	–	Wie oben, für eine Matrix ‚C'
DX	–	Auf zwei Stellen gerundeter Wert des Betrages der Dämpfung D
FX	–	Auf eine Nachkommastelle gerundeter Wert der gerade betrachteten Frequenz
A$	A-String	Zwischenwert für Dialog
A	Akkumulator	Virtueller Akku für Zwischenprüfungen von Benutzer-Anweisungen sowie weitere kurzzeitig benötigte Zwischenwerte
CC	C-Complex	Realteil des Ergebnisses komplexer Divisionen
CI	C-Imaginär	Imaginärteil des Ergebnisses komplexer Divisionen
ZZ	Zähler	Realteil des Dividenden für komplexe Divisionen
ZI	Zähler-Imaginär	Imaginärteil des Zählers
NN	Nenner	Realteil des Divisors für komplexe Divisionen
NI	Nenner-Imaginär	Imaginärteil des Nenners
Z1	Impedanz 1	Speicher für CC
Z2	Impedanz 2	Speicher für CI
G	–	Realteil komplexer Leitwerte
BL	–	Induktiver Leitwert
BC	–	Kapazitiver Leitwert
B	–	Imaginärteil komplexer Leitwerte
R	–	Realteil komplexer Widerstände
XL	–	Induktive Komponente komplexer Widerstände
XC	–	Kapazitive Komponente komplexer Widerstände
X	–	Imaginäre Komponente komplexer Widerstände
DR	Divisor	Nenner für Inversionen
MM	–	Realteil des Faktors 1 für komplexe Multiplikationen
MI	–	Imaginärteil des Faktors 1 für komplexe Multiplikationen
NN	–	Realteil des Faktors 2 für komplexe Multiplikationen
NI	–	Imaginärteil des Faktors 2 für komplexe Multiplikationen
PP	Produkt	Realteil des Ergebnisses komplexer Multiplikationen
PI	Produkt-Imaginär	Imaginärteil des Ergebnisses komplexer Multiplikationen
V, V1...V8	Value, Value N	Temporäre Speicher für Zwischenergebnisse
I, I1...I4	Value N, Imaginärteile	Temporäre Speicher für Zwischenergebnisse
SI	S-Imaginär	Imaginärkomponente zu SS
RI	R-Imaginär	Imaginärkomponente zu RR
DD	Dämpfung	Realteil der Dämpfung
DI	Dämpfung-Imaginär	Imaginäranteil zu DD
D	Dämpfung logarithm.	in Dezibel ausgedrückter Wert des Betrages von DD plus j mal DI

*) siehe Tabelle „Plätze und Werte der Bauteile"

Erweiterung für Wobbel-PET

Das beschriebene Programm ist also fix und fertig zur Berechnung von Filterdämpfungen. Es bietet darüber hinaus alle elementaren Handreichungen, die man benötigt, um sich ein Sonderprogramm nach Maß zu schneidern. Anregungen dafür liegen in der Luft: Wenn man noch andere Zusammenschaltungs-Formen als die hier allein verwendete Kettenschaltung einführt, kann man mit beliebig verzweigten Vierpolen umgehen, durch Einführung aktiver Bauelemente gar ergibt sich eine schier unbegrenzte Anzahl von Variationsmöglichkeiten. Dazu allerdings sollte die angegebene Literatur zu Rate gezogen werden, wenn man noch nicht „vierpolfest" ist.

Eine Bemerkung zum Schluß: Der Autor ärgert sich oft über die unzureichende Dokumentation von BASIC-Programmen, die zwar wunderschön laufen, sich nahezu jeder Änderung aber entziehen, weil es einfach nicht gelingt, den Ablauf genügend transparent zu machen. Grund: unzureichender Begleittext. Hier hat der Autor einmal vorgeführt, wie er sich ein solches Programm dokumentiert wünscht: Und nun: Räumen Sie Ihren Wobbler zur Seite, wobbeln Sie mal mit „Wobbel-PET"!

Literatur

1 *Feldkeller, R.:* „Einführung in die Vierpoltheorie der elektrischen Nachrichtentechnik". Stuttgart 1976. S. Hirzel Verlag.

2 *Harm, H.:* Schaltungsanalyseprogramm für Kleinrechner. ELEKTRONIK 1975, H. 4, S. 101...106, H. 5, S. 83...86, H. 6, S. 78...82.

3 *Hamerli, S.:* Wobbelkurven ohne Meßgerät ermittelt. ELEKTRONIK 1978, H. 5, S. 50...53.

4 Röhre und Transistor als Vierpol. Herausgeber AEG-Telefunken, Fachbereich Röhren, Ulm.

Tabelle der Variablen

Hier werden, in der Reihenfolge ihres ersten Auftretens im Programm, die benutzten Variablen erläutert.

Name	Mnemotechnisches Bezugswort	Funktion
N	–	Schleifen-Zähler
NR	Nummer	Anzahl der Schaltungselemente
SS	–	Realteil der Generator-Impedanz
RR	–	Realteil der Lastimpedanz
FG%(N)	Flag	Schlüssel für Vielfachverzweigung*)
A$(N)	–	Schlüssel für Bauteil-Anordnung und -Art im Netzwerk*)
WR(N)	Wert für R	Ohmsche Komponente des Bauteils N
WL(N)	Wert für L	Induktive Komponente des Bauteils N
WC(N)	Wert für C	Kapazitive Komponente des Bauteils N
F4	Frequenz 4	Ersatzwert für F1
F5	Frequenz 5	Ersatzwert für F2
F6	Frequenz 6	Ersatzwert für F3
F1	Frequenz 1	Startfrequenz der Untersuchung
F2	Frequenz 2	Endfrequenz der Untersuchung
F3	Frequenz 3	Frequenz-Intervall
FF	–	Kreisfrequenz Omega
C1,C3, C5,C7	–	Realanteil der Koeffizienten einer 2x2-Matrix namens ‚C'
C2,C4, C6,C7	–	Imaginäranteil der Koeffizienten der Matrix C
A1...A8	–	Wie oben, für eine Matrix ‚A'
B1...B8	–	Wie oben, für eine Matrix ‚C'
DX	–	Auf zwei Stellen gerundeter Wert des Betrages der Dämpfung D
FX	–	Auf eine Nachkommastelle gerundeter Wert der gerade betrachteten Frequenz
A$	A-String	Zwischenwert für Dialog
A	Akkumulator	Virtueller Akku für Zwischenprüfungen von Benutzer-Anweisungen sowie weitere kurzzeitig benötigte Zwischenwerte
CC	C-Complex	Realteil des Ergebnisses komplexer Divisionen
CI	C-Imaginär	Imaginärteil des Ergebnisses komplexer Divisionen
ZZ	Zähler	Realteil des Dividenden für komplexe Divisionen
ZI	Zähler-Imaginär	Imaginärteil des Zählers
NN	Nenner	Realteil des Divisors für komplexe Divisionen
NI	Nenner-Imaginär	Imaginärteil des Nenners
Z1	Impedanz 1	Speicher für CC
Z2	Impedanz 2	Speicher für CI
G	–	Realteil komplexer Leitwerte
BL	–	Induktiver Leitwert
BC	–	Kapazitiver Leitwert
B	–	Imaginärteil komplexer Leitwerte
R	–	Realteil komplexer Widerstände
XL	–	Induktive Komponente komplexer Widerstände
XC	–	Kapazitive Komponente komplexer Widerstände
X	–	Imaginäre Komponente komplexer Widerstände
DR	Divisor	Nenner für Inversionen
MM	–	Realteil des Faktors 1 für komplexe Multiplikationen
MI	–	Imaginärteil des Faktors 1 für komplexe Multiplikationen
NN	–	Realteil des Faktors 2 für komplexe Multiplikationen
NI	–	Imaginärteil des Faktors 2 für komplexe Multiplikationen
PP	Produkt	Realteil des Ergebnisses komplexer Multiplikationen
PI	Produkt-Imaginär	Imaginärteil des Ergebnisses komplexer Multiplikationen
V V1...V8	Value Value N	Temporäre Speicher für Zwischenergebnisse
I I1...I4	Value N, Imaginärteile	
SI	S-Imaginär	Imaginärkomponente zu SS
RI	R-Imaginär	Imaginärkomponente zu RR
DD	Dämpfung	Realteil der Dämpfung
DI	Dämpfung-Imaginär	Imaginäranteil zu DD
D	Dämpfung logarithm.	in Dezibel ausgedrückter Wert des Betrages von DD plus j mal DI

*) siehe Tabelle „Plätze und Werte der Bauteile"

Plätze und Werte der Bauteile

Die Anordnung der betrachteten Bauteile im Netzwerk, ihre „Plätze" sowie ihre Eigenschaften werden in Datenfeldern gespeichert, die hier näher erläutert sind:

N	Ordnungszahl des Bauelementes. Numerierung beginnt am Netzwerk-Eingang
A$ (N)	Anordnung und Art
WR (N)	Ohmsche Komponente
WL (N)	Induktive Komponente
WC (N)	Kapazitive Komponente
FG% (N)	Integer-Variable als mitlaufendes Flagwort

Anordnung und Art	AS (N)	FG% (N)
Längswiderstand	LR	1
Längsspule	LL	2
Längskondensator	LC	3
Längs-Parallelkreis	LK	4 *)
Querwiderstand	QR	5
Querspule	QL	6
Querkondensator	QC	7
Quer-Serienkreis	QK	8 **)

*) Hierbei ist WR (N) ein hypothetischer Parallelwiderstand RP, der die zusammengefaßten Kreisverluste beschreibt.

**) Hierbei ist WR (N) ein hypothetischer Serienwiderstand RV, der die zusammengefaßten Kreisverluste beschreibt.

Katalog der Programmsegmente

Das Software-Paket ‚Wobbel-PET' gibt dem Hobby-Elektroniker und Profi eine Sammlung von Programm-Segmenten an die Hand, aus denen er sich Dienstprogramme nach Wahl für seine speziellen Anwendungsfälle zusammensetzen kann. Dem folgenden Katalog kommt eine besondere Bedeutung zu: Er nennt Aufruf-Adressen, verwendete Variable und Funktion der Programmsegmente. Wem an schneller Einsatzbereitschaft seines Programmes mehr liegt als an einem vertieften Verständnis des Software-Paketes, der braucht sich um den Inhalt der Segmente fürs erste nicht zu kümmern; es genügt erst einmal, ein Programm zu schreiben, das die aus diesem Katalog ausgewählten Segmente aufruft und verknüpft, ohne Rücksicht auf die Feinstruktur der Segmente, und zwar in Abhängigkeit vom gewünschten Spezialfall. Bei Bedarf ist der Befehl „Return" einzufügen, wo nicht vorhanden.

Aufr.-Adr.	Name	Funktion
16	Start	Löscht Variable und Dimensionierung, bringt dem PET die Zehner-Logarithmen bei.
40	Eingabe 1	Befragt den Operator nach Einzelheiten von Netzwerk und Abschlüssen, dimensioniert Felder und füllt sie.
260	Eingabe 2	Fragt nach Anfangsfrequenz, Schlußfrequenz und Intervall. Wird 0 eingegeben, bleibt der entsprechende Altwert erhalten.
300	Hauptschleife	Anwendungsspezifisches Hauptprogramm, das je nach Anwendungsfall zu ändern ist. Im vorgestellten Beispiel wird eine Tabelle Betriebsdämpfung eines Netzwerkes in Abhängigkeit von der Frequenz geschrieben, also das Zahlenmaterial für eine Filterkurve.
500	Schlußschleife	Bietet die Möglichkeit zur Parameter-Änderung an und fragt, wie's weitergehen soll.
1000	Matrizen-Multiplikation	Schaltet die Bauteile-Gruppe in Matrix A per Kettenschaltung mit der Gruppe in Matrix B zusammen. Eigenschaften des neuen Gebildes in Matrix C.
1100	Matrizentransfer	Kopiert Matrix C in die Matrix B.
1200	Komplexe Division	Teilt die Zahl ZZ plus j mal ZI durch die Zahl NN plus j mal NI, Ergebnis CC plus j mal CI.
1300	Eingangsimpedanz	Errechnet die Eingangsimpedanz eines in Matrix B beschriebenen Netzwerkes in Abhängigkeit von Ausgangsbeschaltung und Kreisfrequenz. Wird im vorgestellten Anwendungsbeispiel nicht aufgerufen, ist aber höchst nützlich für mancherlei Untersuchungen. Übergibt den Realteil dieser Betriebs-Eingangsimpedanz in Z1, den Imaginärteil in Z2. Der Betrag Z3 steht nach Einfügen folgender Zeile zur Verfügung: Z3= SQR(Z1↑2+Z2↑2).
1400	Einzelmatrizen	Acht Subprogramme, die aus den Angaben WR (N), WL (N), WC (N) über die Eigenschaften eines Bauteiles seine Vierpol-Koeffizienten in die Matrix A einlesen (Kettenkoeffizienten). Aufruf über die Zeile ONFG%(N)GOSUB 1400, 1425, 1450, 1475, 1500, 1525, 1550, 1575 Damit sind die Anordnung und Art des Bauelementes mathematisch umgesetzt.
1600	Komplexe Multiplikation	Multipliziert den Faktor 1 (MM plus j mal MI) mit dem Faktor 2 (NN plus j mal NI). Ergebnis: Realteil in PP, Imaginärteil in PI.
1650	Dämpfung 1	Berechnet DD, DI und D, bezogen auf die Leerlaufspannung des Generators.
1800	Dämpfung 2	Wie oben, jedoch bezogen auf die dem Generator maximal entnehmbare Leistung.
2000	Diagnose: Matrizenausdruck	Druckt die Koeffizienten der Matrizen A, B und C, getrennt nach reellen und imaginären Komponenten.
2050	Diagnose: Beträge	Druckt Beträge der besonders kritischen Matrizen-Koeffizienten.
2200	Diagnose: Variable 1	Druckt Frequenzangaben, komplexe Leitwerte, komplexe Widerstände, Angaben zu den Netzwerk-Abschlüssen und Einzelheiten der Betriebsdämpfung (Realteil, Imaginärteil, Tangens des Phasenwinkels).
2300	Diagnose: Variable 2	Druckt einen Katalog von Zwischenergebnissen. Nützlich zur Fehlersuche und zur Entnahme von Zwischenwerten für Laufzeituntersuchungen und anderes.
2400	Prüflauf	Stellt für die weitere Verarbeitung zu Prüfzwecken nach Programmänderungen ein Netzwerk nach Hamerli [3] zur Verfügung, das den Test aller Programmfunktionen erlaubt. Für RV und RP werden Ersatzwerte benutzt, die nahe bei Rechner-Null und bei Rechner-Unendlich liegen. Dies ist erforderlich, um Division durch 0 zu vermeiden: Denn verlustfreie Schwingkreise gibt's ja nicht, der Rechner würde sich mit Fehlermeldung beschweren und anhalten.
10000		Gittermuster-Generator zur Vorbereitung der fotografischen Dokumentation von Rechenergebnissen.

Für keine andere Mikrocomputerschnittstelle sind so viele Funktionseinheiten zu einem derart niedrigen Preis auf dem Markt wie für den sogenannten S-100-Bus. Trotz einiger technischer Unzulänglichkeiten ist dieses Bussystem deshalb nicht nur für den Hobbyisten, sondern auch für den professionellen Entwickler eine interessante Alternative – besonders, wenn spätere Erweiterungen eingeplant sind.

Bernd Pol

Der S-100-Bus – ein de-facto-Standard auf dem Mikrocomputermarkt

1 Geschichte

Im Januar 1975 brachte die amerikanische Firma MITS einen auf Intels Mikroprozessor 8080 basierenden Mikrocomputerbausatz, genannt Altair, zum Preis von 400 $ heraus. Es war der erste für breitere Kreise erschwingliche Mikrocomputer und stieß genau in eine Marktlücke, die aus dem großen Kreis von Elektronik-Begeisterten bestand, die ihre durch Prospekte und einführende Literatur erworbenen theoretischen Kenntnisse in der Praxis erproben wollten. Der Altair wurde, wie man so sagt, ein Bombenerfolg. Er löste die gewaltige Hobbycomputerwelle in den USA aus, deren Ausläufer mittlerweile auch uns erreichen. Angeregt durch den Erfolg des Altair fanden sich rasch weitere Hersteller, die entweder ihren eigenen Computer nach diesem Konzept bauen oder kompatible Systemkarten anbieten. Mittlerweile sind es in den USA über hundert, davon mindestens fünf Hersteller von vollständigen Mikrocomputersystemen. Die Produktionspalette ist außergewöhnlich breit. Sie umfaßt sowohl Standardeinheiten, wie CPU-Karten, Ein/Ausgabeeinheiten oder Speicher (bis zu 64 KByte auf einer Karte!), als auch Karten für Spezialgebiete wie elektronische Spiele, Sprachein- und -ausgabe.

2 Das Konzept

Das Konzept von MITS sieht einen modularen Aufbau vor, bei dem sich die einzelnen Funktionseinheiten (CPU, Speicher, Ein/Ausgabeeinheiten) um einen Systembus gruppieren, über dessen Leitungen die Adressen, Daten und Steuersignale laufen. Für den MITS-Entwurf hat sich die Bezeichnung S-100-Bus

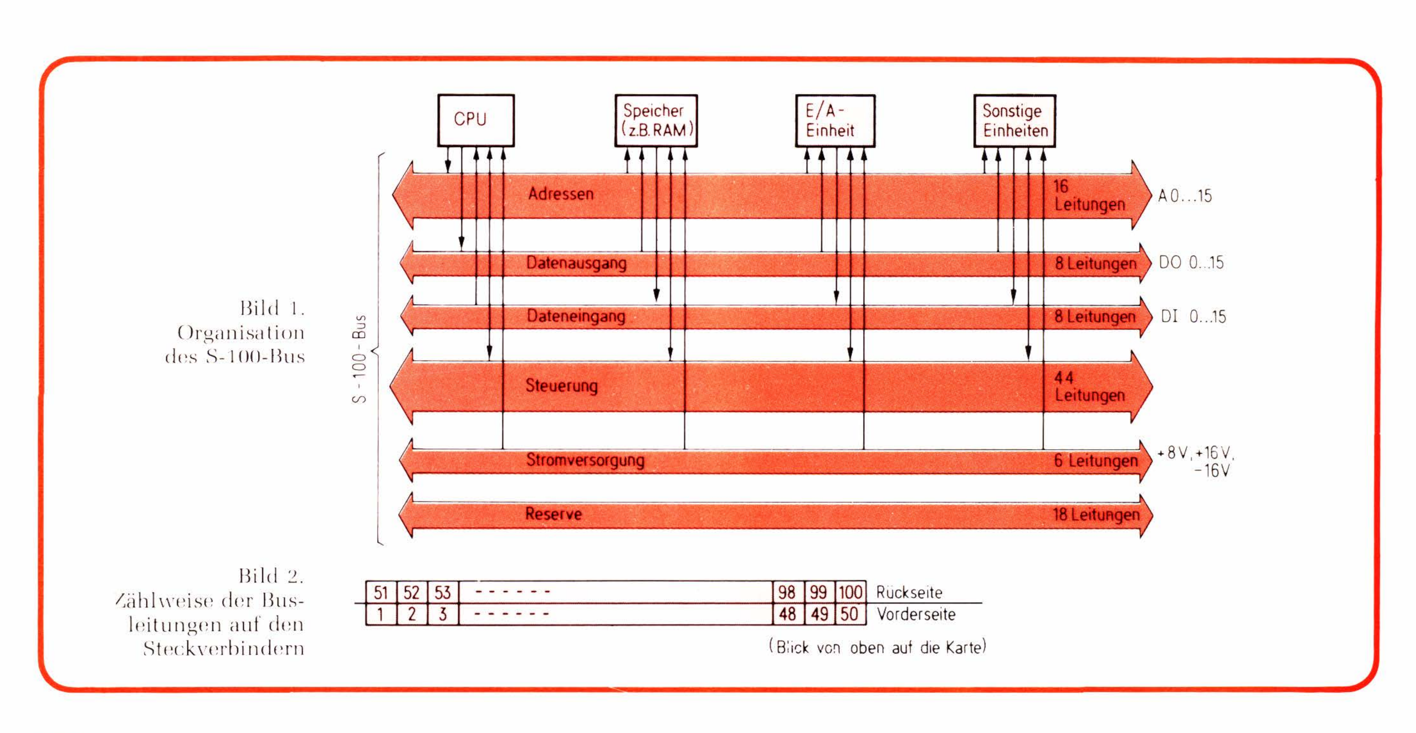


Bild 1. Organisation des S-100-Bus

Bild 2. Zählweise der Busleitungen auf den Steckverbindern

eingebürgert, was „Standard-Einhundert" bedeutet und darauf hinweist, daß 100 Busleitungen zur Verfügung stehen. Diese lassen sich in sechs Gruppen zerlegen (*Bild 1*): Adreßbus, Dateneingangsbus, Datenausgangsbus, Steuerbus, Stromversorgung und Reserveleitungen. Die Systemkarten stehen dabei mit dem Bus über 2 x 50polige Steckverbinder (direktes Stecken) in Kontakt. Die Zählung der Kontakte ist in *Bild 2* wiedergegeben. *Tabelle 1* gibt einen Überblick über die Kontaktbelegung, die im folgenden genauer besprochen wird.

2.1 Unidirektionale Busleitungen

Ein wesentliches Merkmal des Entwurfes ist es, daß alle Busleitungen unidirektional ausgelegt sind. Das heißt die entsprechenden Daten und Signale laufen, bezogen auf die CPU-Karte, auf ihnen nur in einer Richtung: entweder von der CPU zu den anderen Einheiten (Ausgabeleitungen, A) oder von diesen zur CPU (Eingabeleitungen, E).

2.2 Datenleitungen

Wesentliche Konsequenzen hat dies für den Datenteil des S-100-Bus. In den meisten anderen Entwürfen wird dieser bidirektional ausgelegt, dient also der CPU sowohl zum Lesen als auch zum Schreiben von Daten. Die S-100-Konzeption der unidirektionalen Busleitungen bedingt die Aufspaltung des Datenbereiches in zwei Kanäle zu je acht Leitungen: einen Dateneingangsbus (DI 0...7), über den die CPU Daten empfängt, und einen Datenausgangsbus (DO 0...7), über den die CPU Daten zu den anderen Einheiten aussendet. Die Vorteile einer solchen Lösung liegen auf der Hand: Zum einen wird die Zeitsteuerung der Daten vereinfacht – die kritischen Zeiten, in denen die Daten stabil auf dem Bus liegen müssen, werden entschärft; zum anderen ist ein externer Datenzugriff, etwa durch DMA, einfacher möglich; zum dritten sind Rausch- und Übersprecherscheinungen, die durch Ein- und Ausschwingvorgänge auf den Busleitungen beim raschen Wechsel von Schreib- und Lesedaten entstehen, weitgehend ausgeschaltet.

2.3 Steuerleitungen

Auf dem Bus ist eine auffallend große Zahl von Steuerleitungen (44 Stück) definiert. Sie ermöglichen einen ausgedehnten Verkehr der verschiedenen Einheiten mit der CPU, Abwicklung von Interrupts auf acht Ebenen und externen Zugriff auf die Systemkarten unter Umgehung der CPU. Funktionell lassen sie sich in acht Gruppen zusammenfassen:

2.3.1 Taktleitungen

Diese vier Leitungen gliedern sich in zwei Untergruppen (*Tabelle 2*). Die erste dient unmittelbar zur Synchronisation der CPU mit den anderen Einheiten. Sie trägt die Taktsignale $\Phi 1$ und $\Phi 2$ des Prozessors, die (gepuffert) aus dem Taktgenerator abgeleitet werden. Nicht bei allen CPU-Entwürfen werden beide Taktsignale $\Phi 1$ und $\Phi 2$ auf den Bus geführt. Der Grund liegt in der Verwendung integrierter Taktgeneratoren (z. B. 8224), die nur $\Phi 2$ in TTL-Pegel liefern. Da manche Systemkarten beide Taktsignale benötigen, kann dies unter Umständen zu Funktionsstörungen führen. Die zweite Gruppe hat allgemeinere Aufgaben. Zum einen verteilt sie das (gepufferte) Primärsignal des Quarzoszillators (bei 8080A-Prozessoren meist 2 MHz), das den externen Einheiten bei Bedarf zum Erzeugen interner Signale dient. Zum anderen wird über die RTC-Leitung ein Echtzeitsignal verteilt.

Tabelle 1. Anschlußbelegung beim S-100-Bus

Anschluß	Richtung*	Bezeichnung	Anschluß	Richtung*	Bezeichnung
01		+ 8 V	48	A	SHLTA
02		+16 V	49	A	CLOCK
03	E	XRDY	50		Masse
04	(E)	VI 0	51		+ 8 V
05	(E)	VI 1	52		–16 V
06	(E)	VI 2	53	E	$\overline{\text{SSW DSB}}$
07	(E)	VI 3	54	A	$\overline{\text{EXT CLR}}$
08	(E)	VI 4	55	(A)	RTC
09	(E)	VI 5	56...67		frei
10	(E)	VI 6	68	A	MWRT
11	(E)	VI 7	69	(E)	$\overline{\text{PS}}$
12...17		frei	70	(A)	PROT
18	E	$\overline{\text{STA DSB}}$	71	(A)	RUN
19	E	$\overline{\text{C/C DSB}}$	72	E	PRDY
20	(A)	UNPROT	73	E	$\overline{\text{PINT}}$
21	(A)	SS	74	E	$\overline{\text{PHOLD}}$
22	E	ADD DSB	75	E	$\overline{\text{PRESET}}$
23	E	DO DSB	76	A	PSYNC
24	A	$\Phi 2$	77	A	$\overline{\text{PWR}}$
25		$\Phi 1$	78	A	PDBIN
26	A	PHLDA	79	A	A0
27	A	PWAIT	80	A	A1
28	A	PINTE	81	A	A2
29	A (A)	A5	82	A	A6
30	A	A4	83	A	A7
31	A	A3	84	A	A8
32	A	A15	85	A	A13
33	A	A12	86	A	A14
34	A	A9	87	A	A11
35	A	DO 1	88	A	DO 2
36	A	DO 0	89	A	DO 3
37	A	A10	90	A	DO 7
38	A	DO 4	91	E	DI 4
39	A	DO 5	92	E	DI 5
40	A	DO 6	93	E	DI 6
41	E	DI 2	94	E	DI 1
42	E	DI 3	95	E	DI 0
43	E	DI 7	96	A	SINTA
44	A	SM1	97	A	$\overline{\text{SWO}}$
45	A	SOUT	98	A	SSTACK
46	A	SINP	99	E	$\overline{\text{POC}}$
47	A	SMEMR	100		Masse

* von der CPU-Karte aus gesehen: E = Eingang, A = Ausgang
Angaben in Klammern: Leitung wird nur bei besonderen CPU-Entwürfen auf die Karte geführt.

Tabelle 2. Taktleitungen

Anschluß	Bezeichnung	Bedeutung
25	$\Phi 1$ phase 1 clock	Taktsignal, Phase 1
24	$\Phi 2$ phase 2 clock	Taktsignal, Phase 2
49	CLOCK chrystal clock	Taktsignal vom Quarzoszillator
55	RTC real time clock	Echtzeitsignal

Dieses wird in der Regel von einer speziellen Systemkarte erzeugt.

2.3.2 Direkte Eingänge zum Prozessor

Hierbei handelt es sich um die beiden Ready-Signale, die die Bereitschaft der Systemkarten zum Datenaustausch mit der CPU anzeigen, und um die Unterbrechungsanforderungen Interrupt-Request und Hold-Request (*Tabelle 3*). Die beiden Ready-Signale PRDY aus dem System selbst und XRDY von externen Einheiten werden auf der CPU-Karte UND-verknüpft und an Anschluß 23 des 8080-Prozessors geführt, während die Unterbrechungsanforderungen (über Invertierer) unmittelbar mit den Kontakten 13 (HOLD) bzw. 14 (INT) verbunden sind.

2.3.3 Befehls- und Steuersignale des Prozessors

Diese Gruppe von Signalen wird direkt vom Prozessor erzeugt und an dessen Anschlüssen 16 (INTE), 17 (DBIN), 18 ($\overline{WR}$), 19 (SYNC), 21 (HLDA) und 24 (WAIT) ausgegeben. Die Signale werden auf der CPU-Karte gepuffert und dann auf den Systembus gegeben. Mit ihnen synchronisiert die CPU den Datenverkehr mit den Systemkarten (*Tabelle 4*).

2.3.4 Statusleitungen

Ebenfalls zur Synchronisation des Datenverkehrs gibt der Prozessor am Anfang jedes Maschinenzyklus ein sogenanntes Statuswort auf den Datenbus. Dieses Statuswort wird im S-100-System bereits auf der CPU-Karte decodiert und in einem 8-bit-Register gespeichert, dessen Ausgänge über Puffer ständig mit dem Bus verbunden sind. Dadurch liegt die Statusinformation während des gesamten Maschinenzyklus auf den mit dem Präfix S bezeichneten Steuerleitungen an (*Tabelle 5*). Die Folge ist eine Entlastung des Datenausgangsbusses und eine Vereinfachung im Entwurf der übrigen Systemkarten, die den CPU-Status nicht jedesmal neu zu decodieren brauchen.

2.3.5 Speichersteuerung

Von diesen vier Leitungen (*Tabelle 6*) braucht nur eine (MWRT, *Memory Write Enable*) unbedingt zur CPU-Karte zu führen. Die drei anderen können auch von der Bedienerkonsole oder einer speziellen Systemkarte benutzt werden. Es handelt sich dabei um die Möglichkeit, einen bestimmten Bereich im RAM-Speicher gegen Überschreiben zu schützen. Dazu muß allerdings die betreffende Speicherkarte entsprechend vorbereitet sein. Ein ausgewählter Speicherblock (Schritte von 256 Byte...4 KByte Tiefe sind üblich) wird geschützt, indem die Anfangsadresse des zu schützenden Blocks auf den Adreßbus gegeben wird und gleichzeitig die PROT-Leitung (*Protect*, Schützen) auf H-Potential gehoben wird. Dadurch wird ein RS-Flipflop auf der Speicherkarte gesetzt, dessen Ausgang über ein Gatter den Schreibimpuls MWRT blockiert (*Bild 3*). Entsprechend hebt man einen Speicherschutz mit der UNPROT-Leitung wieder auf.

Tabelle 3. Direkte Prozessoreingänge

Anschluß	Logik	Bezeichnung		Bedeutung
72	pos.	PRDY	processor ready	Bereit-Signal von Systemeinheiten
3	pos.	XRDY	external ready	Bereit-Signal von externen Einheiten
74	neg.	$\overline{PHOLD}$	processor hold request	Unterbrechungsanforderung, bewirkt Anhalten des Prozessors
73	neg.	$\overline{PINT}$	processor interrupt request	Unterbrechungsanforderung, bewirkt Abarbeiten des Interrupt-Programms

Tabelle 4. Befehls- und Steuersignale des Prozessors

Anschluß	Logik	Bezeichnung		Bedeutung
28	pos.	PINTE	processor interrupt enable	Signalisiert, daß das „Interrupt-Enable"-Flipflop gesetzt ist und der Prozessor Unterbrechungsanforderungen entgegennimmt
78	pos.	PDBIN	processor data bus in	gibt an, daß der Datenbus frei zur Aufnahme von Daten aus der adressierten Einheit ist
77	neg.	$\overline{PWR}$	processor writes	gibt an, daß die vom Prozessor ausgegebenen Daten stabil sind
76	pos.	PSYNC	processor synchronize	gibt den Beginn jedes Maschinenzyklus an
26	pos.	PHLDA	processor hold acknowledge	signalisiert, daß die CPU angehalten (im HOLD-Status) ist
27	pos.	PWAIT	processor wait acknowledge	signalisiert einen Wartezyklus des Prozessors

Tabelle 5. Statusleitungen

Anschluß	Logik	Bezeichnung		Bedeutung
96	pos.	SINTA	status bit 0: interrupt acknowledge	gibt an, daß eine Unterbrechungsanforderung empfangen wurde
97	neg.	SWO	status bit 1: write operation	gibt an, daß eine Schreiboperation ausgeführt wird
98	pos.	SSTACK	status bit 2: stack adressed	gibt an, daß der Inhalt des Stapelzeigers auf dem Adreßbus liegt
48	pos.	SHLTA	status bit 3: halt acknowledge	gibt an, daß ein HLT-Befehl empfangen wurde
45	pos.	SOUT	status bit 4: output device addressed	gibt an, daß der Adreßbus die Adresse einer Ausgabeeinheit trägt
44	pos.	SM1	status bit 5: machine cycle 1	signalisiert den Beginn eines Befehlszyklus (Maschinenzyklus 1)
46	pos.	SINP	status bit 6: input device addressed	gibt an, daß der Adreßbus eine Eingabeeinheit adressiert
47	pos.	SMEMR	status bit 7: memory read cycle	gibt an, daß ein Speicher-Lese-Zyklus vollzogen wird

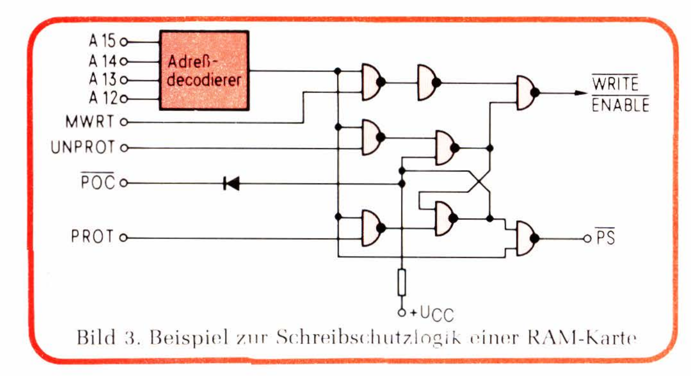


Bild 3. Beispiel zur Schreibschutzlogik einer RAM-Karte

Tabelle 6. Speichersteuerleitungen

Anschluß	Logik	Bezeichnung		Bedeutung
68	pos.	MWRT	memory write enable	gibt einen Schreibzyklus des Prozessors an, Schreibimpuls für die Speicher
70	pos.	PROT	protect memory area	Leitung, um einen Speicherbereich gegen Überschreiben zu schützen
20	pos.	UNPROT	unprotect memory area	Leitung, um einen Schreibschutz aufzuheben
69	neg.	$\overline{PS}$	protect status	gibt an, daß der adressierte Speicherbereich geschützt ist

Tabelle 7. Externer Zugriff

Anschluß	Logik	Bezeichnung		Bedeutung
18	neg.	$\overline{STADSB}$	status buffer disable	schaltet die Status-Puffer auf der CPU-Karte ab
19	neg.	$\overline{C/CDSB}$	command/control buffer disable	schaltet die Befehls- und Steuerpuffer auf der CPU-Karte ab
22	neg.	$\overline{ADDDSB}$	address buffer disable	schaltet die Adreß-Puffer auf der CPU-Karte ab
23	neg.	$\overline{DODSB}$	data out buffer disable	schaltet die Datenausgangs-Puffer auf der CPU-Karte ab
53	neg.	$\overline{SSWDSB}$	sense switch buffer disable	schaltet die Dateneingangs-Puffer auf der CPU-Karte ab

Tabelle 8. Rücksetz- und Statusleitungen

Anschluß	Logik	Bezeichnung		Bedeutung
99	neg.	$\overline{POC}$	power on clear	initialisiert das System beim Einschalten der Stromversorgung
75	neg.	$\overline{PRESET}$	processor reset	Rücksetzsignal für den Prozessor und entsprechende Logik auf der CPU-Karte
54	neg.	$\overline{EXTCLR}$	external clear	Signal zum Löschen von E/A-Einheiten
21	pos.	SS	single step mode	gibt an, daß das Programm in Einzelschritten abgearbeitet wird
71	pos.	RUN	run mode	gibt an, daß das Programm abgearbeitet wird

Wird ein geschützter Speicher adressiert, so wird dies durch die $\overline{PS}$-Leitung (meist zur Konsole) signalisiert. Das System wird beim Einschalten so initialisiert, daß jeder Speicherschutz aufgehoben ist.

2.3.6 Leitungen für externen Zugriff

Die beiden Datenbusse, der Adreßbus und die Steuerleitungen aus den Tabellen 4 und 5 werden auf der CPU-Karte über Puffer mit Tristate-Ausgängen geführt. Diese Puffer lassen sich über die fünf Steuerleitungen dieser Gruppe in den hochohmigen Zustand schalten (*Tabelle 7*), womit auf die zugehörigen Busleitungen von außen zugegriffen werden kann. Der jeweils angesteuerte Puffer geht aus der Bezeichnung der Steuerleitungen hervor. Im Fall $\overline{SSW DSB}$ werden die Dateneingangspuffer auf der CPU-Karte abgeschaltet, wodurch z. B. von der Konsole mit den Datenschaltern (bei Altair: *Sense Switches*) Daten in das System eingegeben werden können.

2.3.7 Interruptleitungen

Diese Leitungen führen in der Regel nicht zur CPU-Karte. Mit ihnen werden verschiedene Unterbrechungsebenen (*Vectored Interrupts*) angesteuert, die von einer gesonderten Einheit decodiert werden müssen. Diese ist meistens auf einer eigenen Systemkarte untergebracht und erzeugt bei Vorliegen der richtigen Priorität eine Unterbrechungsanforderung auf der PINT-Leitung.

2.3.8 Sonstige Steuerleitungen

Hier lassen sich zwei Untergruppen angeben (*Tabelle 8*). Die eine führt zum Initialisieren des gesamten Systems beim Einschalten ($\overline{POC}$), zum Rücksetzen der CPU während des normalen Betriebs ($\overline{PRESET}$) oder zum Löschen von E/A-Einheiten ($\overline{EXT\ CLR}$). Die andere besteht aus zwei Signalleitungen, die in erster Linie von der Konsole in Anspruch genommen werden, nämlich zur Anzeige von Einzelschritt- (*Single Step*, SS) und Normalabarbeitung (RUN) eines Programms. Sie führen meist nicht zur CPU-Karte, sondern geben den Status entsprechender Flipflops auf der Konsole an.

2.4 Pegel und Lastfaktoren

Alle signalführenden Busleitungen haben TTL-Pegel. Dabei wird in der Regel positive Logik (1 = H, 0 = L), bei einigen Steuerleitungen auch negative Logik (1 = L, 0 = H) verwendet. In den Tabellen 2...8 ist die verwendete Logik angegeben. Die meisten Leitungen können über die Puffer auf den Karten bis zu 10 Standard-TTL- bzw. 30 Low-Power-Schottky-TTL-Eingänge treiben. Die Eingänge in die Karten dürfen maximal einer Standard-TTL-Last entsprechen, so daß die Lastfaktoren für ein voll ausgebautes System (je nach Ausführung bis zu 25 Karten) in der Regel ausreichen.

2.5 Stromversorgung

Das Konzept sieht Spannungsregler auf den einzelnen Systemkarten vor. Dadurch werden unerwünschte Kopplungen über die Versorgungsleitungen weitgehend ausgeschaltet. Außerdem genügt so eine einfache unstabilisierte zentrale Stromversorgung. Sie liefert an den Bus folgende Spannungen:

+ 8 V an Leitung 1 und 51,
+16 V an Leitung 2,
–16 V an Leitung 52,
Masse an Leitung 50 und 100.

Die 8-V-Versorgung wird auf der Karte auf die Standardspannung +5 V heruntergeregelt und dient in erster Linie zur Versorgung der TTL-Schaltungen. Sie wird für jede Systemkarte benötigt. Die beiden anderen Spannungen dienen in erster Linie zur Versorgung von MOS-Schaltungen. Auch sie werden auf der Karte auf die erforderliche Größe heruntergeregelt. Ein voll ausgebautes System benötigt etwa +8 V/10...20 A, +16 V/2...5 A und –16 V/1 A.

2.6 Reserveleitungen

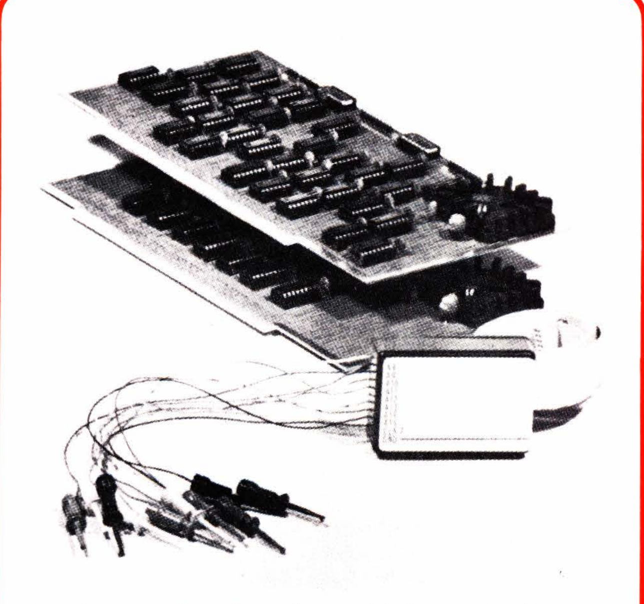
Stromversorgung, Adreß-, Daten- und Steuerleitungen belegen zusammen 82 der 100 Busleitungen. Die übrigen 18 Leitungen bleiben für Spezialanwendungen und Systemänderungen frei. Es gibt jedoch bereits Vorschläge zur Standardisierung für 16-bit-Mikrocomputer (auf der Basis des TMS 9900 von Texas Instruments) und für neue Speichermedien wie CCD- oder Magnetblasenspeicher, die zusätzliche Steuerleitungen benötigen.

3 Diskussion

Den idealen Systembus gibt es (noch) nicht, und es hat folglich eine ganze Reihe von Alternativentwicklungen zum S-100-Bus gegeben, die dessen Schwächen umgehen sollten. Einige der schwachen Seiten des Konzeptes seien hier aufgezählt:

– Aus der Übersicht in Tabelle 1 geht hervor, daß Adreß- und Datenleitungen unmittelbar benachbart sind. Das führt zum störenden *Übersprechen* beider, was vor allem eine Erhöhung des Grundrauschens zur Folge hat, das nicht immer vernachlässigt werden kann. Entsprechendes gilt für die beiden Taktleitungen Φ1 und Φ2. Abhilfe bringt hier die Einführung eines Masseschirmes zwischen den Blöcken, wie sie von einigen Herstellern der den Bus tragenden Mutterplatinen (*Motherboards*) bereits angeboten wird.

– Das Konzept der verteilten Spannungsregelung hat den Nachteil, daß die *Wärmebelastung* zwischen den Karten oft beträchtlich ansteigt. Das kann bis zum Ausfall einzelner Teile führen. Beim Systemaufbau muß daher sorgfältiger als bei anderen Entwürfen auf gute Wärmeableitung durch Konvektion geachtet werden. Dies gilt um so mehr, als die Kühlkörper wegen des geringen Platinenabstandes recht klein gehalten werden müssen. Schon relativ kleine Systeme benötigen deshalb für sicheren Betrieb einen Ventilator.



Für den S-100-Bus gibt es die verschiedensten Steckkarten: Das Bild zeigt z. B. einen Logikanalysator für 24 Kanäle (Databyte)

– Der S-100-Bus ist durch die Tatsache, daß die Steuersignale und das Statuswort des 8080-Prozessors direkt auf dem Bus liegen, etwas überspezialisiert. Will man die aus der großen Herstellerzahl resultierenden ökonomischen und technischen Vorteile auch für andere Mikroprozessoren (z. B. 6800, 6502) oder Weiterentwicklungen des 8080-Prozessors (z. B. Z 80) nutzen, so sind oft aufwendige *Anpaßschaltungen* an das abweichende Steuersystem und den Zeitablauf notwendig (dessenungeachtet gibt es für die genannten Prozessoren bereits Entwicklungen für den S-100-Bus).

4 Ausblick

Der S-100-Bus hat sich durchgesetzt, und zwar in dem Maße, daß einige Hersteller zwei Versionen ihrer Entwicklungen anbieten, eine für den Intel-(oder hauseigenen)Bus und eine für den S-100-Bus. Man wird trotz aller Nachteile des Entwurfs daher auch in Zukunft und auch in Europa, speziell in der Bundesrepublik, mit dem S-100-Standard rechnen müssen. Das hat verschiedene Gründe:

– die Fülle des Angebots für dieses Konzept, wie sie schon erwähnt worden ist;

– die Tatsache, daß die starke Konkurrenz vor allem kleiner und mittlerer Hersteller naturgemäß die Preise drückt (eben diese Konkurrenz bewirkt auch, daß ständig Produkte auf hohem und technisch fortgeschrittenstem Niveau auf dem Markt erscheinen);

– last not least ist die Tatsache nicht zu übersehen, daß mit dem Mikroprozessor 8080 auf einem Industriestandard aufgebaut wurde, der sich nach Expertenaussage möglicherweise zwei Jahrzehnte halten wird.

Literatur

1 *Zaks, R.*: Microprocessors, Sybex Inc., Berkeley, Cal. S. 302...304.

Mikrocomputer-Lerngerät und industrielles Modulsystem aus einer Hand

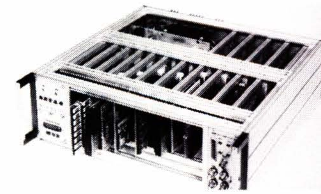
Die Einarbeitung in die Mikrocomputertechnik ist nur anhand eigener Programmierpraxis möglich. Die Arbeit mit den üblichen Kits, die zu diesem Zweck angeboten werden, krankt nicht selten an der mangelnden Dokumentation in deutscher Sprache und an der fehlenden Möglichkeit, das System für industrielle Zwecke weiter ausbauen zu können. Die Berliner Firma MCS will mit einem vielversprechenden Konzept diese Mängel ausräumen:



Sie bietet ein nach kommerziellen Maßstäben aufgebautes System an, dessen Grundelemente das Lern- und Entwicklungshilfsgerät ALPHA 1 und das für den industriellen Einsatz konzipierte Modulsystem BETA 8 sind. ALPHA 1 ist ein netzanschlußfertiges Gerät. Beide Modelle sind mit dem Mikroprozessor 6502 aufgebaut und uneingeschränkt miteinander kompatibel. Sie haben eine V-24-, eine Strom- und eine TTL-Schnittstelle, verfügen über einen Taktgenerator für Lochstreifeneingabe und über die Anschlußmöglichkeit für zwei Kassettenrecorder. Sofern der Recorder über eine Start/Stopp-Steuerung verfügt, kann auch diese Betriebsart vom Programm benutzt werden. Um bereits mit ALPHA 1 kleine Steuerungsprobleme zu lösen, stehen dem Anwender zusätzlich 16 freidefinierbare Ein/Ausgabekanäle zur Verfügung. Alle Ein- bzw. Ausgabeinformationen sind an entsprechende Steckverbindungen geführt, die sich an der Frontplatte befinden.

Sowohl ALPHA 1 wie auch BETA 8 sind mit einem eigenen von MCS entwickelten Betriebssystem ausgestattet. Die Struktur des 2-K-Monitorprogrammes MONA ist so ausgelegt, daß ein großer Teil dieser Betriebssoftware in Anwenderprogramme miteinbezogen werden kann. Sämtliche Ein- und Ausgabeelemente werden durch das Betriebssystem gesteuert. Als Eingabeelemente stehen dem Anwender beim Modell ALPHA 1 eine sedezimale Tastatur nebst elf speziellen Funktionstasten sowie eine 8stellige Siebensegmentanzeige für die Ausgabeinformation zur Verfügung. Durch die softwaremäßige Ansteuerung lassen sich auch Buchstaben zur Anzeige bringen. Diese Möglichkeit wird in dem ebenfalls vorhandenen Disassembler ausgenutzt, der eine mnemonische Darstellung des Befehlssatzes erlaubt. Die internen Register des Mikroprozessors lassen sich per Tastendruck aufrufen. Weitere Betriebsarten sind „Einzelschritt" und „langsamer Programmablauf". Anstelle von Tastatur und Anzeigeeinheit ist auch die Kommunikation mit einem Terminal möglich. Zur Unterstützung ist von den Entwicklern ein etwa 100seitiges Handbuch zu ALPHA 1 herausgebracht worden. Außer der Beschreibung des Systems sind sämtliche Schaltunterlagen und das Assemblerlisting des Betriebssystems MONA vorhanden. Für Schulungs- und Lehrzwecke stehen ebenfalls technische Unterlagen in Form von Overheadfolien zur Verfügung. Auf besonderen Wunsch veranstaltet MCS auch Schulungsseminare.

Das System BETA 8 besteht aus 16 Europakarten, die in beliebiger Konfiguration eingesetzt werden können. Es wird im Grundaufbau im 19-Zoll-Tischgehäuse oder bei weiterem Ausbau im Einschubgehäuse untergebracht. Für das System stehen in der ersten Ausbaustufe alle wichtigen Karten zur Verfügung. Im einzelnen handelt es sich hierbei um die CPU-Karte, ausgerüstet mit dem Mikroprozessor 6502, eine 2-K-RAM-Karte, eine 8-K-RAM-Karte, eine universelle bis maximal 8K bestückbare



Speicherkarte für eine große Anzahl von ROMs, PROMs bzw. EROMs. Ferner stehen zur Verfügung eine E/A-Karte mit zwei PIAs, eine universelle Experimentierkarte mit einem 40poligen Sockel einschließlich einer in 8-K-Stufen möglichen Adreßcodierung, eine Interfacekarte zum Anschluß für das Mini-Floppy-Disk-Laufwerk von BASF sowie eine Anzahl von verschiedenen Zwischenadaptern u. a. zum Betrieb des BETA-8-Systems mit einer Eingabe- und Anzeigekonsole sowie in Kombination mit dem ALPHA-1-Mikrocomputer.

☐ Hersteller: MCS Micronic Computer Systeme GmbH, Tennstedter Str. 18, 1000 Berlin 46, Tel. (0 30) 7 11 30 11

Während Mikroprozessoren und Mikrocomputer in den letzten Jahren immer billiger wurden, blieben die Peripheriegeräte relativ teuer. In letzter Zeit werden aber Datensichtgeräte angeboten, die einen Fernsehempfänger als Display benutzen und in einer vernünftigen Preisrelation zu Mikrocomputern stehen. Eines dieser Ein/Ausgabegeräte ist das Video-Terminal AT-64, das von der Firma Astronic (München) als kompletter Bausatz (ohne Monitor) für 990 DM + MwSt. angeboten wird.

Lothar Scheider

Ein/Ausgabegerät für Mikrocomputer

Eine Übersicht der Eigenschaften des Geräts gibt die *Tabelle*. Der Lieferumfang des Bausatzes umfaßt Gehäuse, Chassis, Tastatur, Leiterplatten, alle aktiven und passiven Bauelemente, Kleinteile und eine umfangreiche Bauanleitung (z. Zt. in Englisch), so daß zum Zusammenbau nur etwas Geschick, ein Lötkolben und etwa zehn Stunden Zeit notwendig sind. Erleichternd kommt hinzu, daß alle integrierten Schaltungen in einer Richtung orientiert sind und Funktionen wie Tastatur, Ein-/Ausgabe, Zeichenspeicher und Cursor-Steuerung auf separaten Karten enthalten sind, die auf die Hauptplatine aufgesteckt werden, so daß der abschließende Funktionstest schrittweise durchgeführt werden kann.

Bild 1 zeigt die Blockschaltung des Geräts, das vorwiegend mit TTL-Bausteinen aufgebaut ist, aber auch drei MOS-LSI-Bausteine enthält.

Funktion

Auf dem Bildschirm werden 32 bzw. 64 Zeichen pro Zeile und 16 Zeilen pro Seite dargestellt, wobei jedes Zeichen aus einer Matrix von 7 x 9 Punkten besteht. Die Zeichen werden voneinander durch eine Leerspalte, die Zeichenzeilen durch sieben Leerzeilen getrennt. Hinzu kommen 16 Zeilen für den Bildsynchronimpuls und weitere 40 Leerzeilen, so daß insgesamt 312 Zeilen entstehen, die der deutschen Fernsehnorm (312 1/2 Zeilen pro Halbbild) weitgehend entsprechen.

Außer dem Bildsignal benötigt der Monitor noch Zeilen- und Bildsynchronimpulse. Impulsbasis ist eine PLL-Schaltung, die mit der Netzfrequenz von 50 Hz getriggert wird und über eine Frequenzmultiplikation um den Faktor 312 einen Ausgangsimpuls mit einer Impulsfolgefrequenz von 15 600 Hz liefert, die der Horizontalablenkfrequenz von 15 625 Hz eines Video-Monitors bzw. Fernsehers sehr nahe kommt und eine einwandfreie Synchronisation gewährleistet. Vom Zeilensynchronimpuls wird ein Verzögerungsimpuls abgeleitet, der eine seitliche Justierung des Monitorbildes ermöglicht. Der Aufbau des Monitorbildes entsteht durch das Zusammenspiel von Punktzähler (3 bit ≙ 8 Punkte pro Zeichen), Zeichenzähler (5 bit ≙ 32 Zeichen pro Zeile bzw. 6 bit ≙ 64 Zeichen pro Zeile), Zeilenzähler (4 bit ≙ 16 Zeilen pro Zeichenzeile) und Zeichenzeilenzähler (4 bit ≙ 16 Zeichenzeilen pro Bild).

Der erwähnte Verzögerungsimpuls setzt Punkt- und Zeichenzähler zurück und erhöht den Zeilenzähler sowie im Übertragsfall den Zeichenzeilenzähler. Parallel zum Zeichenzeilenzähler läuft der Aufrollzähler, der neben dem Seiten- wahlweise den Aufrollmodus zuläßt. Der Zeilenzähler erzeugt auch den Bildsynchronimpuls. Der Punktoszillator taktet sowohl den Punktzähler als auch das Punktschieberegister, das die Zeicheninformation parallel vom Zeichengenerator übernimmt und seriell ausgibt. Kombiniert mit Zei-

Tabelle. Eigenschaften des Video-Terminals AT-64

64 oder 32 Zeichen pro Zeile
16 Zeilen pro Seite
Seiten- oder Aufrollmodus
Interner Speicher für 1024 Zeichen
128 Zeichen (ASCII)
Groß- und Kleinbuchstaben
Umschaltbar zwischen weißen Zeichen vor schwarzem und schwarzen Zeichen vor weißem Hintergrund
Steuerbarer Cursor
Summer für Zeilenende
V-24-Schnittstelle mit einstellbarer Übertragungsgeschwindigkeit (110, 150, 300, 600, 1200 Bd)
BAS-Signal-Ausgang (als Option mit UHF-Modulator zum Anschluß an den Antenneneingang eines Fernsehempfängers)

len- und Bildsynchronimpuls ergibt sich daraus das BAS-Signal für den Monitor.

Der Zeichengenerator hat insgesamt elf Adreßeingänge, von denen sieben, die Zeicheneingänge, durch die im Speicher enthaltenen ASCII-Zeichen angesteuert werden, während die restlichen vier, die Zeileneingänge, vom Zeilenzähler gesteuert werden.

Beim Drücken einer Taste erzeugt der Tastendecodierer ein entsprechendes ASCII-Signal, das über den später beschriebenen UART in den 8-bit-Zwischenspeicher übernommen wird. Der Hauptspeicher selbst wird fortlaufend adressiert, da das Monitorbild 50mal pro Sekunde aufgefrischt werden muß, so daß die Übernahme vom Zwischen- in den Hauptspeicher erst mit der Adresse erfolgen kann, durch die das erste freie Zeichenfeld, das gleichzeitig auch die Cursor-Position ist, definiert wird. Die fortlaufende Speicheradresse wird vom Zeichenzähler und Aufrollzähler erzeugt, während der Cursor einen eigenen 10-bit-Positionszähler besitzt, der vom Tastenübernahmeimpuls getaktet wird.

Die Zählerstände werden über einen 10-bit-Komparator verglichen, der bei Koinzidenz einen Ausgangsimpuls zur Übernahme des Zeichens in den Hauptspeicher erzeugt. Der Cursor selbst wird durch Invertierung des jeweiligen Positionsfeldes erzeugt und kann durch Cursor-Steuerbefehle, die den 10-bit-Cursor-Zähler ändern, beliebig innerhalb des Bildfeldes verschoben werden. Das Interface mit dem Computer wird vom bereits erwähnten UART (Universeller Asynchroner Empfänger/Sender) gebildet, der senderseitig die parallelen 7-bit-ASCII-Zeichen in serielle Zeichen mit Start- und Stoppbit umwandelt, die mit anschließender Pegelumwandlung V-24-kompatibel sind. Empfangene serielle Zeichen werden in parallele rückgewandelt.

Eine detaillierte Funktionsbeschreibung ist in der jedem Bausatz beiliegenden Bauanleitung enthalten. Der Bausatz ist in Deutschland bereits in größeren Stückzahlen verkauft worden, wobei sich keine Funktions- oder Zuverlässigkeitsprobleme ergeben haben. *Bild 2* zeigt das Ein/Ausgabegerät mit einem Fernsehempfänger. *Lothar Scheider*



Bild 2. Video-Terminal VT-64 mit Fernsehempfänger

Bild 1. Blockschaltung des Video-Terminals VT-64

Dipl.-Ing. Rainer Müller

Prinzip und Arbeitsweise von Floppy-Disk-Speichern

In den vergangenen Jahren traten die Floppy-Disk-Speicher einen unaufhaltsamen Siegeszug an. Dieser Trend wurde ausgelöst durch den Wunsch nach einem billigen, schnellen und zuverlässigen Massenspeicher für kleine Rechnersysteme und die Tatsache, daß die Entwicklung von Festkörper-Speichern mit ähnlichen Eigenschaften noch nicht abzusehen war. Der Floppy-Disk-Speicher ist ein magnetisch arbeitender Speicher, der zwischen herkömmlichen Plattenspeichern und Magnetbandspeichern eingeordnet werden kann. Die Speicherplatten, die sich in quadratischen Schutzhüllen befinden, sind sehr einfach zu handhaben und erlauben einen unkomplizierten und schnellen Programm- und Datenaustausch. Außer in Mikrocomputersystemen werden den Floppy-Disk-Speicher in Datenerfassungsanlagen, intelligenten Terminals, Textverarbeitungssystemen und vielen anderen Bereichen eingesetzt.

1 Aufbau und Organisation von Floppy-Disk-Speichern

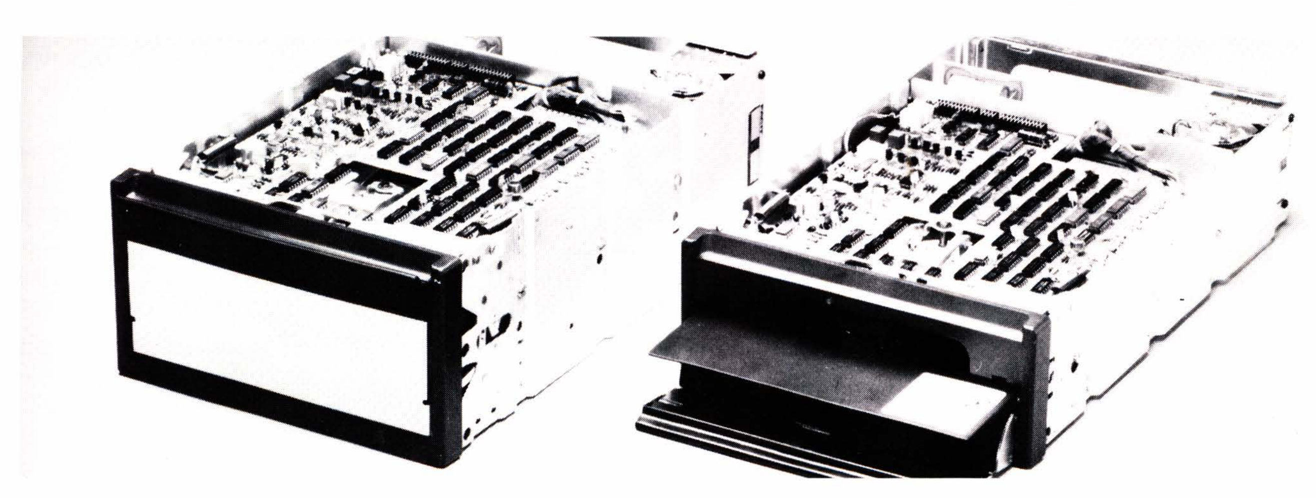
Ein Floppy-Disk-Speicher ist aufgebaut aus dem Laufwerk (*floppy disk drive*), in das die Speicherplatte (Diskette) eingebracht wird, und der Steuerung (*floppy disk controller*), die in der Regel mehrere Laufwerke bedienen kann. Floppy-Disk-Speicher gibt es in verschiedenen Ausführungsformen, die nach der Größe der verwendeten Disketten eingeteilt werden können:

- Standard-Disketten (Umhüllung 20,3 x 20,3 cm^2), die seit einigen Jahren genormt sind, werden in sogenannten Standard-Floppy-Disk-Speichern verwendet.
- Mini-Disketten (Umhüllung 13 cm x 13 cm^2), für die noch keine Norm besteht, werden seit kurzem in sogenannten Mini-Floppy-Disk-Speichern verwendet.
- Euro-Disketten (Umhüllung 10,5 cm x 10,5 cm^2) wurden für spezielle Anwendungen entwickelt und sind zur Normung vorgeschlagen.

Standard- und Mini-Floppy-Disk-Speicher haben prinzipiell die gleiche Arbeitsweise. Sie unterscheiden sich in der Kapazität und den Geräteabmessungen. Ein von der Firma SEL entwickelter Floppy-Disk-Speicher mit Euro-Disketten arbeitet nach teilweise anderen Verfahren. Seine Bedeutung wird er vor allem bei der Anwendung in Endgeräten der Textkommunikation erlangen. Er soll hier jedoch nicht näher beschrieben werden. Die folgenden detaillierten Ausführungen beziehen sich nur auf die Standard-Ausführung.

1.1 Das Speichermedium

Der Datenträger eines Standard-Floppy-Disk-Speichers ist eine flexible Kunststoffplatte von 20 cm Durchmesser, die mit einer nichtorientierten magnetischen Oxydschicht versehen ist. Zur besseren Handhabung ist die Platte fest umschlossen von einer quadratischen Schutzhülle, in der sie auch während des Betriebes verbleibt. In der Platte ist ein Loch für die Zentrierung und den Plattenantrieb und das Index-Loch. Die Hülle hat drei Öffnungen: eine für den Plattenantrieb, eine für die Abfrage des Index-Loches in



der Platte und eine lange ovale für den radial bewegten Schreib-Lese-Kopf (*Bild 1*).

Die Datenspeicherung erfolgt überlicherweise auf 77 konzentrischen, 0,3 mm breiten Aufzeichnungsspuren mit der Spur 0 als Außenspur. Die Spurdichte beträgt dabei 1,89 Spuren/mm oder 48 tpi (*tracks per inch*). Die gesamte Speicherkapazität einer Plattenseite wird bei gleicher Speicherkapazität aller Spuren von der möglichen Bitdichte auf der innersten, also kürzesten Spur, der möglichen Spurdichte und dem verwendeten Aufzeichnungsverfahren bestimmt.

Die erzielbare Bitdichte hängt davon ab, wie gut man aufeinanderfolgende Flußwechsel des magnetischen Speichermaterials beim Lesen auflösen kann. Bei heutigen Systemen liegt die Flußwechseldichte bei 257 Flußwechsel/mm oder 6520 fci (*flux changes per inch*). Die mögliche Spurdichte hängt hauptsächlich von der erreichbaren Positioniergenauigkeit des Magnetkopfes und von physikalischen Eigenschaften des Plattenmaterials ab (z. B. Dehnung durch Luftfeuchtigkeit). So werden bei einer Aufzeichnung mit Wechseltaktschrift, die von den gebräuchlichen Verfahren die geringste Speicherdichte ergibt und die noch erläutert wird, 3,2 Mbit Speicherkapazität pro Plattenseite erreicht. Von der vorhandenen Speicherkapazität wird ein großer Teil für die Speicherorganisation (Formatierung) benötigt, so daß z. B. mit dem Standard-Format IBM 3740 nur noch 1,9 Mbit an nutzbarem Speicherplatz zur Verfügung stehen.

Da während des Schreib-Lese-Vorgangs die Platte fest gegen den Magnetkopf gedrückt wird, unterliegt die magnetische Beschichtung einer gewisser Abnützung. Nach Herstellerangaben liegt die Lebensdauer einer Diskette bei $2 \cdot 10^6 ... 5 \cdot 10^6$ Abfragen/Spur.

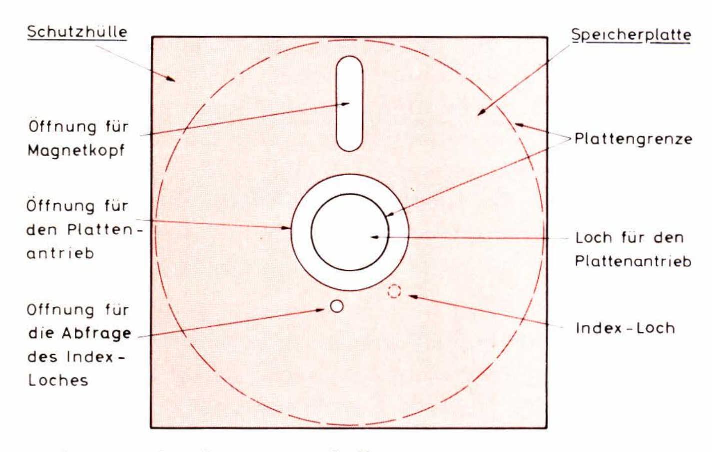


Bild 1. Speicherplatte mit Umhüllung

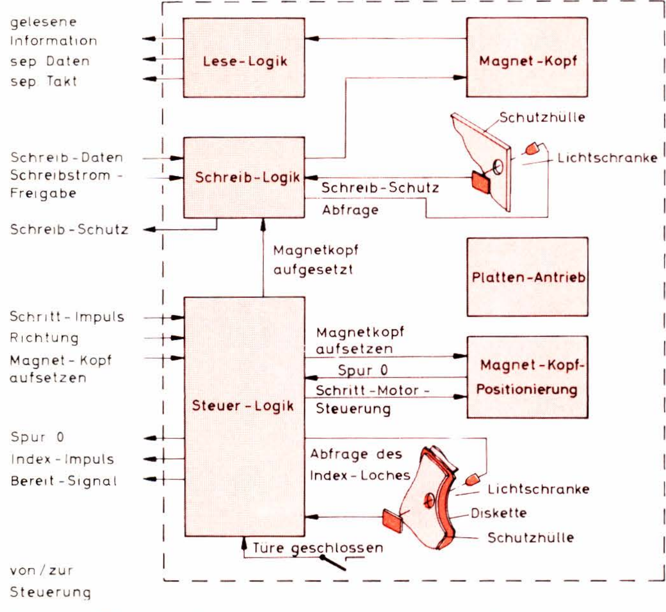


Bild 2. Blockschaltung eines Floppy-Disk-Laufwerkes

1.2 *Das Laufwerk*

Das Floppy-Disk-Laufwerk enthält den Antrieb für die Diskette, den Schreib-Lese-Kopf mit Positionierungseinrichtung und Steuerlogik, die Schreib-Lese-Logik sowie Lichtschranken für die Abfrage des Index-Loches und Kontrollfunktionen. Das Laufwerk muß der Steuerung anzeigen, wann es arbeitsbereit ist, und es muß auf deren Befehle und Anforderungen reagieren. Die Blockschaltung eines Laufwerkes zeigt *Bild 2*.

Durch den Plattenantrieb wird die Diskette auf eine Rotationsgeschwindigkeit von 360 U/min gebracht. Die Geschwindigkeit wird in der Regel über die Abfrage des Index-Loches kontrolliert. Als Zeitnormal kann ein der Schreibtakterzeugung dienender Quarzoszillator benutzt werden. Hat die Diskette die richtige Geschwindigkeit und ist die Türe geschlossen, dann wird dieser Zustand durch das Bereit-Signal (*ready signal*) der Steuerung gemeldet.

Zum Schreiben oder Lesen wird der Magnetkopf von der Magnetkopf-Positionierung auf die der gewünschten Aufzeichnungsspur entsprechende Position gebracht. Er ist auf einem beweglichen Schlitten montiert, der bei den meisten Systemen auf einer Präzisionsspindel läuft. Diese kann durch einen Schrittmotor gedreht werden, wobei ein Umdrehungsschritt des Motors der Bewegung des Magnetkopfes von einer zur benachbarten Spur entspricht (Spurwechsel). Die Schrittimpulse und ein Signal für die Bewegungsrichtung des Magnetkopfes werden von der Steuerung vorgegeben.

Gebräuchliche Laufwerke benötigen für einen Spurwechsel etwa 3...10 ms. Für das Aufsetzen des Magnetkopfes sind noch etwa 30 ms zu berücksichtigen. Die mittlere Zugriffszeit liegt bei 490 ms. Der tatsächliche Wert hängt davon ab, von welcher Position aus eine neue Spur gesucht werden soll. Steht der Magnetkopf über der Spur 0, auf deren Position er auch beim Einlegen einer Diskette steht, dann wird dies der Steuerung automatisch angezeigt. Dieses Signal dient als Bezugspunkt für die Bewegung des Magnetkopfes.

Zur Durchführung einer Schreib-Lese-Operation wird der Magnetkopf, nachdem er in die erforderliche Position gebracht worden ist, durch den Befehl „Magnetkopf aufsetzen" gegen die Speicherplatte gedrückt.

Zum Schreiben werden die Schreibdaten von der Steuerung an die Schreiblogik gesendet und von ihr entsprechend der verwendeten Aufzeichnungsart in Stromimpulse umgewandelt, die dem Magnetkopf nach Anliegen des Signals „Schreibstrom-Freigabe" (*write enable*) zugeführt werden. Während des Schreibvorganges werden eventuell vorhandene alte Aufzeichnungen überschrieben. Wird nach dem IBM-3740-Format aufgezeichnet, dann wird üblicherweise gleichzeitig mit spurabhängiger Stromamplitude gearbeitet; ab Spur 43 aufwärts wird der Schreibstrom vermindert, um bei kleinerer Länge der Aufzeichnungsspur noch die erforderliche Bitdichte zu erreichen.

Beim Lesen werden die vom Magnetkopf aufgenommenen Signale zunächst verstärkt, dann digitalisiert und der Leselogik zugeführt. An deren Ausgängen stehen neben der gelesenen Information, die Daten- und Taktimpulse gemischt enthält, auch die separierten Daten- und Taktimpulse zur Weiterleitung an die Steuerung zur Verfügung.

Das Laufwerk enthält in der Regel auch eine Schutzeinrichtung gegen unbeabsichtigtes Einspeichern (und damit Löschen des vorherigen Speicherinhalts) in Form einer Lichtschranke, die eine bestimmte Stelle der Schutzhülle abfragt. Wird dort ein Loch angebracht, dann erhält die Steuerung ein Schreib-Schutz-Signal (*write protect*), das die Bildung des Signales „Schreibstrom-Freigabe" unterbindet. Außerdem wird die Schreiblogik verriegelt.

Die maximale Bitfolgedichte beim Einspeichern bzw. Ausspeichern hängt von den verwendeten Aufzeichnungsverfahren, die unter Abschnitt 1.4 dargestellt werden, ab. Bei Verwendung der Wechseltaktschrift für die Aufzeichnung liegt die Bitfolgedichte bei 250 kbit/s und bei 500 kbit/s mit der modifizierten Wechseltaktschrift und dem M^2FM- und GCR-Code.

1.3 *Die Steuerung*

Als Bindeglied zwischen Rechner und Laufwerk ist eine Steuereinheit erforderlich, die Floppy-Disk-Steuerung. Durch sie erfolgt die Einteilung des Speicherplatzes entsprechend der verwendeten Formatierung, das Aufbereiten der Daten in die für die Speicherung benötigte Form und die Bildung von Prüf-Bytes zur Fehlererkennung. Die Steuerung muß in der Lage sein, die vom zu bedienenden Rechner geforderten Operationen selbständig auszuführen.

Eine Floppy-Disk-Steuerung kann sehr unterschiedlich aufgebaut sein, was von ihrem Verwendungszweck bzw. vom Rechnersystem, mit dem sie zusammenarbeiten soll, abhängt. Es werden sogenannte ROM-Steuerungen, also Einheiten, deren Ablauf durch das in einem ROM abgelegte Programm gesteuert wird, Steuerungen auf Mikroprozessorbasis oder neuerdings solche, die mit speziellen hochintegrierten Schaltungen (LSI) arbeiten, verwendet. Die Steuerung wird direkt mit dem Systembus des Rechners verbunden, mit dem sie zusammenarbeitet und kann in der Regel maximal vier bzw. acht Laufwerke wahlweise bedienen.

Die Datenaufnahme einer Steuerung erfolgt prinzipiell nach folgendem Schema: Die vom Rechner byteweise an ein Datenregister übermittelten Daten werden zur Parallel-Serien-Wandlung in ein Schieberegister gegeben. Liegt ein Schreibbefehl vor und kommt vom Laufwerk die Meldung, daß der Magnetkopf positioniert ist, dann werden die Daten aus dem Schieberegister geschoben und dabei entsprechend der verwendeten Codierung verarbeitet und seriell an das Laufwerk weitergegeben. Die Datenausgabe erfolgt prinzipiell auf dem umgekehrten Weg. Die gelesenen Informationen, die durch die Laufwerklogik digitalisiert wurden, werden decodiert. Die daraus gewonnenen Daten werden in ihrer seriellen Form ins Schieberegister geschoben, von wo aus sie über das Datenregister byteweise parallel abgerufen werden.

Die Steuerung kann verschiedene Operationen des Laufwerkes veranlassen, z. B. automatische Spursuche oder automatisches Abheben des Magnetkopfes nach einer gewissen Anzahl von Leerläufen der Diskette (z. B. ausgelöst durch einen Schreibbefehl, dem keine Schreibdaten folgen).

Zur Fehlererkennung beim Lesen werden beim Schreiben von der Steuerung aus der aufzuzeichnenden Information blockweise Prüf-Bytes nach dem CRC-Verfahren (*Cyclic Redundancy Check*) [1] erzeugt und mit den Daten aufgezeichnet. Beim Lesen werden aus der gelesenen Information nach dem gleichen Verfahren Prüf-Bytes gebildet und mit den aufgezeichneten verglichen. Bei einem Unterschied wird ein wiederholtes Lesen der Information veranlaßt. Wird nach einer gewissen Anzahl von Leseversuchen keine Übereinstimmung erreicht, dann wird der Vorgang abgebrochen und eine Fehlermeldung abgegeben.

Die seit kurzer Zeit auf dem Markt befindlichen Floppy-Disk-Steuerungen auf der Basis von LSI-Schaltungen erlauben es, mit einem Minimum an zusätzlichem Aufwand sämtliche Steuerfunktionen durchzuführen. Sie sind so ausgelegt, daß sie nach dem IBM-3740-Aufzeichnungsformat arbeiten. Jedoch besteht bei einigen die Möglichkeit, mit anderen Formaten zu arbeiten. Um die Anpassung an das verwendete Laufwerk zu optimieren, können teilweise auch die Spurwechseldauer und die Einstellzeit des Magnetkopfes programmiert werden.

1.4 *Die Speicherorganisation*

Um die auf der Diskette gespeicherten Informationen schnell und sicher wiederzufinden, muß die Speicherung nach einem bestimmten festgelegten Prinzip erfolgen. Die einfachste Form wäre eine spurweise Aufteilung des Speicherplatzes. Da jedoch die Speicherkapazität einer Spur für die meisten Anwendungen zu groß ist, wird der Kreisumfang in Sektoren aufgeteilt. Hierfür gibt es zwei Verfahren, das sogenannte Soft- und Hard-Sectoring, d. h. ein Aufteilen durch entsprechende Sektormarken innerhalb der Aufzeichnung bzw. durch, dem Index-Loch vergleichbare, zusätzliche Sektorlöcher auf dem Kreisumfang.

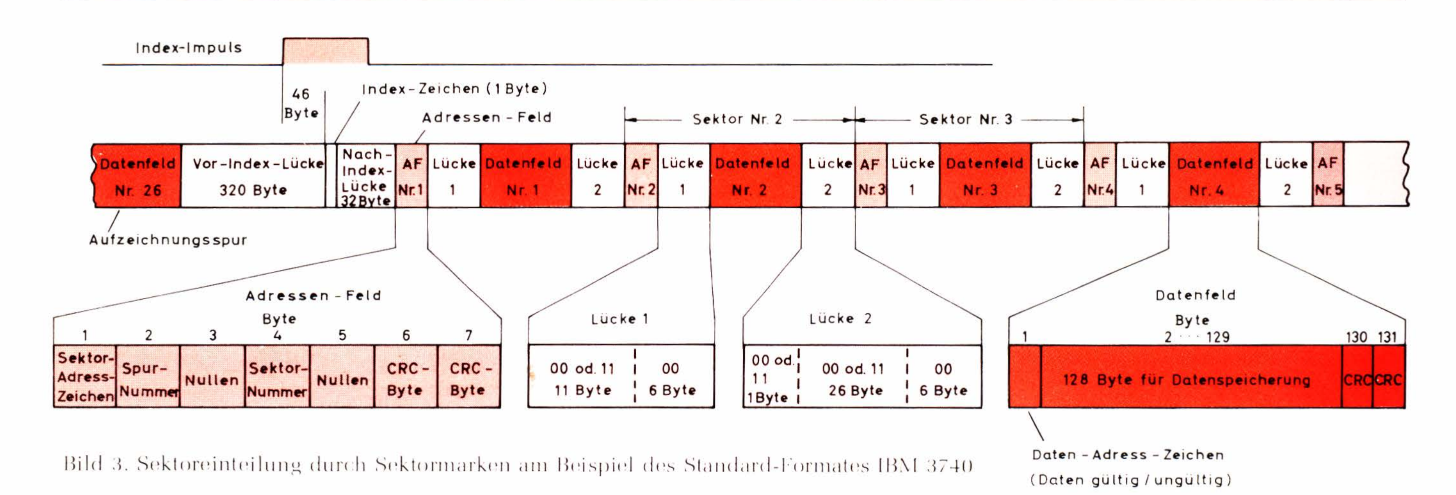


Bild 3. Sektoreinteilung durch Sektormarken am Beispiel des Standard-Formates IBM 3740

Bei der Aufteilung durch Sektormarken muß jede Diskette vor der ersten Anwendung in einem Plattendurchlauf formatiert werden. Dabei wird jede Aufzeichnungsspur in eine bestimmte Anzahl von Sektoren aufgeteilt, an deren Anfang eine Kennung geschrieben wird. Für die spätere Datenspeicherung wird ein bestimmter Teil eines Sektors reserviert, der üblicherweise durch Lücken, die aus Nullen oder Einsen bestehen, abgegrenzt wird. Zur Kennzeichnung des jeweils ersten Sektors der Spuren und damit des Spuranfangs, dient das Index-Loch in der Platte. Es wird mit einer Lichtschranke abgefragt und bildet so durch den Index-Impuls einen festen Bezugspunkt für die Formatierung (siehe Abschnitt 1.2)

Für das genannte Standard-Format ist die Sektoraufteilung in *Bild 3* gezeigt. Dieses Format arbeitet mit 26 Sektoren pro Umfang. Bei der Formatierung wird an den Spuranfang das sogenannte Index-Zeichen geschrieben. Weiterhin werden pro Sektor zur Kennzeichnung des Adressenfeldes das Sektor-Adreß-Zeichen und zur Kennzeichnung des Datenfeldes das Daten-Adreß-Zeichen geschrieben. Diese Zeichen (1 Byte) haben ein festes und durch fehlende Taktimpulse unverwechselbares Bitmuster und dienen zum Identifizieren der Aufzeichnungen. Die Byte-Synchronisation beim Lesen kann durch das Index-Zeichen während des Index-Impulses erfolgen oder auch über die Adreß-Zeichen.

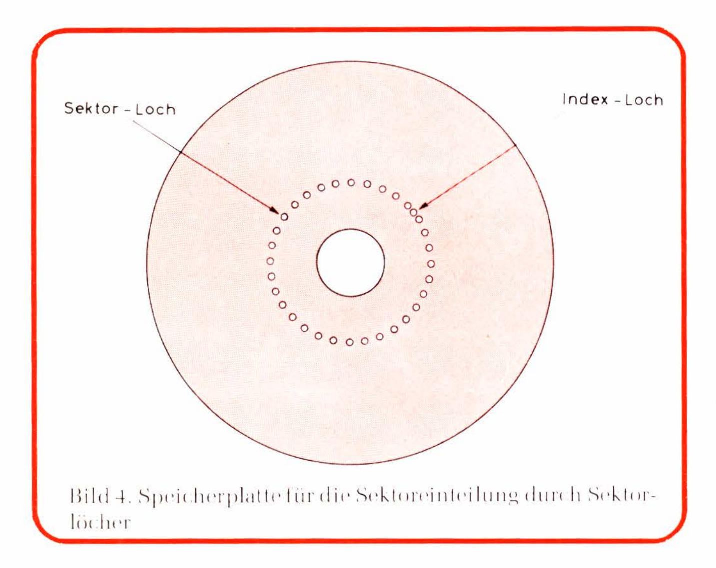


Bild 4. Speicherplatte für die Sektoreinteilung durch Sektorlöcher

Neben diesem Standard-Format sind auch noch andere Formate möglich. Durch Vergrößern des Datenfeldes (durch Reduktion der Sektorzahl oder Verkleinern der Lücken) oder durch variable Sektorlänge (Sektorlänge an Datenblocklänge angepaßt) läßt sich die effektive Speicherkapazität erhöhen. Die Verwendung solcher Sonderformate hat jedoch den Nachteil, daß beschriebene Disketten zwischen verschiedenen Systemen in der Regel nicht mehr ausgetauscht werden können, wie bei Verwendung des Standard-Formates. Außerdem ist eine eigene Formatierungseinrichtung erforderlich, die für das Standard-Format nicht unbedingt benötigt wird, da entsprechend formatierte Disketten auf dem Markt zur Verfügung stehen.

Bei der Einteilung durch Sektorlöcher ist die Diskette neben dem Index-Loch mit 32 weiteren Löchern versehen (*Bild 4*). Diese kennzeichnen durch die mit Hilfe einer Lichtschranke erzeugten Sektorimpulse den Anfang der bei diesem Verfahren gebräuchlichen 32 Sektoren. Der Spuranfang wird auch hier durch den Index-Impuls markiert. Die Datenblocklänge eines Sektors ist maximal 132 Byte, woraus sich bei 77 Spuren eine Kapazität von 325 KByte pro Plattenseite ergibt.

Die Datenaufzeichnung innerhalb der Sektoren erfolgt sowohl bei der Aufteilung durch Sektormarken als auch durch Sektorlöcher nach einem festen Code, der in der Regel neben den Daten auch den Takt in geeigneter Weise mit aufzeichnet. Am gebräuchlichsten ist bisher die Wechseltaktschrift (FM-Code, Frequency Modulation). Die Aufzeichnung der Information erfolgt in sogenannten Bit-Zellen. Eine Bit-Zelle entspricht hier der Periode zwischen zwei aufeinanderfolgenden Vorderflanken des mitaufgezeichneten Taktes, der zur Synchronisation beim Lesen verwendet wird. Eine gespeicherte Eins wird durch einen Impuls in der Taktpause dargestellt, bei einer Null fehlt dieser Impuls (*Bild 5a*).

Infolge der kontinuierlichen Aufzeichnung des Taktes ist die Wechseltaktschrift nicht sehr effizient. Eine Verbesserung bringt die modifizierte Wechseltaktschrift (*Modified FM-Code*, MFM), auch Miller-Code genannt, bei der spezielle Taktimpulse nur noch bei aufeinanderfolgenden Nullen aufgezeichnet wer-

den (*Bild 5b*). Da die Taktbits eigentlich nur benötigt werden, um die Leseschaltung auch bei einer Serie von Nullen zu synchronisieren, ist dieses Verfahren ausreichend. So wird bei gleichen technologischen Grenzen eine größere Speicherdichte auf der Diskette möglich (Verdoppelung der Speicherdichte auf der Platte und der Bitfolgedichte beim Schreiben bzw. Lesen). Die modifizierte Wechseltaktschrift erfordert allerdings eine komplexere Logik sowohl für ihre Bildung als auch für die Entschlüsselung.

Eine Modifizierung des MFM-Codes ist der M²FM-Code, der, bezogen auf die Wechseltaktschrift, ebenfalls eine Verdoppelung der Speicherdichte ermöglicht. Dieser Code arbeitet im Prinzip wie der MFM-Code, nur wird hier bei einer Serie von Nullen jedes zweite Taktbit nicht mit aufgezeichnet (*Bild 5c*).

Eine Verdoppelung der Speicherdichte wird auch mit der Gruppen-Codierung (GCR, *Group Code Recording*) [2] erreicht. Bei diesem Verfahren wird z. B. jeweils einer Gruppe von vier Bits der Daten eine 5-bit-Kombination zugeordnet, in der nie mehr als zwei Nullen aufeinanderfolgen und an deren Beginn und Ende nur einzelne Nullen stehen (in den 32 Kombinationen der fünf Bits sind 17 Kombinationen der geforderten Art enthalten). Aufgezeichnet werden müssen nur die Einsen, da durch das Vermeiden von längeren „Null-Serien" eine gute Synchronisation beim Lesen gewährleistet ist. Die *Tabelle* zeigt die gebräuchliche Zuordnung der Gruppen.

Ein weiteres Codierungsverfahren ist die bei der Firma SEL entwickelte 3-Längen-Schrift [3], bei der den Bit-Folgen Null-Eins der 1,5fache und Null-Null der 2fache Flußwechselabstand gegenüber der Eins zugeordnet wird. Mit diesem Verfahren lassen sich etwas größere Speicherdichten als mit der modifizierten Wechseltaktschrift erreichen.

2 Operationen eines Floppy-Disk-Speichers

Floppy-Disk-Speicher können auf Anforderung des Rechners verschiedene Operationen, wie das Suchen einer Aufzeichnungsspur und das Schreiben und Lesen der Daten, durchführen. Alle Operationen werden vom Floppy-Disk-Speicher (Laufwerk und Steuerung) nach Eingang eines entsprechenden Befehls selbständig durchgeführt. Um die gespeicherten Daten abrufen zu können, muß der Rechner wissen, wo diese auf der Platte abgelegt sind. Dazu wird beim Beschreiben der Speicherplatte ein Verzeichnis angelegt. Dort wird unter dem Namen des Datenblocks (*file name*) festgehalten, auf welcher Spur und in welchen Sektoren ein gespeicherter Datenblock (*file*) abgelegt ist. Die Ablage dieses Verzeichnisses kann z. B. auf der Spur 0 einer Speicherplatte erfolgen. Bei der Suche nach einem bestimmten Datenblock muß dann zunächst diese Spur gelesen werden, um die Angaben über den Ablageort zu erhalten. Die folgenden Operationen werden so beschrieben, wie sie im typischen Fall ablaufen können.

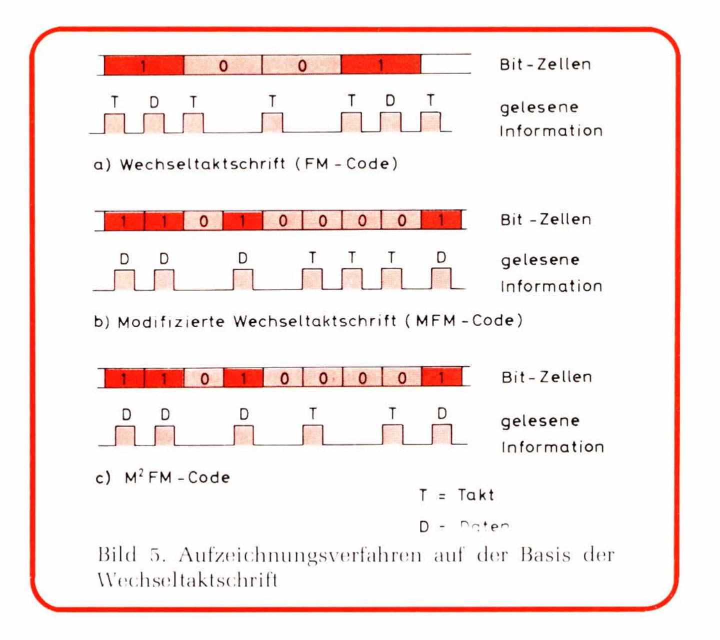


Bild 5. Aufzeichnungsverfahren auf der Basis der Wechseltaktschrift

2.1 *Suchen einer Aufzeichnungsspur*

Zur Vorbereitung eines Lese- oder Schreibvorganges muß der Magnetkopf zunächst auf die Position der dafür vorgesehenen Aufzeichnungsspur gebracht werden. Vom Rechner wird dazu die Nummer der gewünschten Spur an die Steuerung gesendet und dort mit dem Inhalt des Spur-Registers verglichen, in dem die Spurnummer der augenblicklichen Kopfposition steht. Das Ergebnis dieses Vergleiches bestimmt die erforderliche Bewegungsrichtung des Magnetkopfes. Die entsprechende Information wird an das Laufwerk gegeben. Unter gleichzeitigem Verändern der im Spur-Register stehenden Nummer werden dann Impulse für den Schrittmotor der Positionierung abgegeben, bis die im Spur-Register stehende Nummer der gewünschten Aufzeichnungsspur entspricht. Das Ende des Suchvorganges wird der Steuerung angezeigt.

Tabelle. Zuordnung der Bit-Kombinationen bei der Gruppencodierung

4-bit-Gruppen der Daten	5-bit-Kombinationen der Aufzeichnung
0000	11001
0001	11011
0010	10010
0011	10011
0100	11101
0101	10101
0110	10110
0111	10111
1000	11010
1001	01001
1010	01010
1011	01011
1100	11110
1101	01101
1110	01110
1111	01111

Beim Einlegen einer neuen Speicherplatte wird der Magnetkopf automatisch auf die Spur 0 gestellt. Diese Position wird immer der Steuerung gemeldet, dabei wird das Spur-Register, das beim Einlegen auf einem beliebigen Wert stehen kann, auf Null gesetzt.

2.2 Lesen der Daten

Soll, nachdem sich der Magnetkopf über der gewünschten Spur befindet, die in einem Sektor aufgezeichnete Information gelesen werden, dann müssen zunächst die Spur- und Sektornummern in das Spur- und Sektor-Register und der Lesebefehl in das Befehlsregister der Steuerung gegeben werden. Ausgelöst durch den Lesebefehl, wird der Magnetkopf gegen die Platte gedrückt und ein Belegtsignal erzeugt. Zur Kontrolle, ob die richtige Spur gefunden ist, wird die im ersten gelesenen Adressenfeld stehende Spurnummer mit dem Inhalt des Spur-Registers verglichen. Anschließend wird die in diesem Adressenfeld stehende Sektornummer mit dem Inhalt des Sektor-Registers verglichen. Bei fehlender Übereinstimmung wird die nächste Sektornummer gelesen. Stimmen die Spur- und Sektornummern mit den vorgegebenen überein, und ist die CRC-Prüfung für das Adressenfeld positiv verlaufen, dann ist der gesuchte Datenbereich gefunden. In der Regel muß die Steuerung das Adressenfeld mit der richtigen Spur- und Sektornummer sowie die CRC-Bytes innerhalb einer bestimmten Anzahl von Plattenumdrehungen gefunden haben. Bei erfolgloser Suche wird die Befehlsausführung abgebrochen und eine Fehlermeldung abgegeben.

Im zugehörigen Datenfeld muß zunächst das Daten-Adreß-Zeichen gefunden werden. Falls das nicht gelingt, wird der Befehl abgebrochen und signalisiert, daß die Aufzeichnung nicht gefunden wurde. Wurde es gefunden, dann werden die Daten gelesen und nach der Digitalisierung in ein Schieberegister geschoben. Ist dieses gefüllt (1 Byte), dann wird das Byte parallel ins Datenregister gegeben und gleichzeitig dem Rechner dieser Zustand angezeigt. Die folgenden Bytes des Datenfeldes werden entsprechend ausgelesen, bis das gesamte Datenfeld gelesen ist. Hat der Rechner das Datenregister bis zum Einschreiben des nächsten Byte nicht abgefragt, dann geht dieses Byte verloren. Der Verlust wird angezeigt. Wird am Ende des Datenfeldes ein CRC-Fehler festgestellt (beim IBM-Format stehen am Ende des Datenfeldes zwei CRC-Bytes), dann wird das gemeldet und die Befehlsausführung abgebrochen.

2.3 Schreiben der Daten

Zur Vorbereitung des Schreibvorganges muß der Magnetkopf zunächst in die gewünschte Position gebracht werden. Kommt der Schreibbefehl, dann wird der Magnetkopf gegen die Platte gedrückt und der Belegtzustand signalisiert. Nun wird, wie beim Lesevorgang, die Spurnummer geprüft und danach der gewünschte Sektor gesucht. Ist er gefunden, dann werden die Schreib-Daten angefordert. Nachdem das erste Byte ins Datenregister geschrieben ist, wird mit einem bestimmten Abstand zu den CRC-Bytes des Adressenfeldes (Lücke 1 in Bild 3, 11 Byte beim IBM-3740-Format) der Schreibstrom freigegeben. Wird die erste Datenanforderung nicht bedient, dann wird die Befehlsausführung abgebrochen und Datenverlust signalisiert. Ist das Datenregister beschrieben, dann wird entsprechend dem verwendeten Format ein Daten-Adreß-Zeichen geschrieben. Anschließend wird das Datenfeld beschrieben, indem die Steuerung byteweise Daten anfordert, die vom Datenregister in ein Schieberegister gegeben werden, von wo aus sie seriell zum Schreiben ausgegeben werden. Eine neue Datenanforderung erfolgt immer, wenn das Schieberegister leer ist. Wird eine Datenanforderung nicht rechtzeitig bedient, dann wird Datenverlust signalisiert und ein Byte Nullen eingeschrieben. Der Schreibvorgang wird dabei in der Regel nicht unterbrochen. Nachdem das letzte Daten-Byte geschrieben ist, werden die in der Steuerung gebildeten zwei CRC-Bytes ans Ende des Datenfeldes geschrieben. Anschließend wird der Schreibstrom abgeschaltet.

3 Ausblick

Die Entwicklung der Floppy-Disk-Speicher ist noch nicht abgeschlossen. So kamen Mitte 1977 die ersten Floppy-Disk-Speicher für zweiseitiges Beschreiben der Disketten auf den Markt, was bei Verwendung eines Aufzeichnungscodes mit doppelter Speicherdichte eine Vervierfachung der Speicherkapazität pro Diskette gegenüber dem einseitigen Beschreiben mit FM-Code ergibt. Neuerdings wird auch an einer Verdoppelung der Spurenzahl pro Plattenseite gearbeitet. Ebenso ist auch die Entwicklung von Floppy-Disk-Speichern mit kleineren Abmessungen und entsprechend kleineren Disketten noch nicht abgeschlossen.

Literatur

1 *Swanson, R.*: Understanding Cyclic Redundancy Codes. Computer Design Bd. 14 (1975), H. 11, S. 93...99.

2 *Sidhu, P. S.*: Group-Coded Recording Reliably Doubles Diskette Capacity. Computer Design Bd. 15 (1976), H. 12, S. 84...88.

3 *Rein, W. H.*: Optimale Schreibverfahren für magnetische Speicher. Elektrisches Nachrichtenwesen 52 (1977), H. 3, S. 218...222

4 *Kolk, A. J.*: Low-Cost Rotating Memories: Status and Future. COMPUTER March 1976, S. 30...34.

Dipl.-Ing. Rainer Müller, geboren in Kitzingen, studierte an der Technischen Hochschule Darmstadt und ist seit 1969 bei der Standard Elektrik Lorenz AG



(SEL) in Stuttgart tätig. Dort befaßt er sich im Erzeugnisgebiet Fernsprechtechnik mit Fragen der Anwendung integrierter Digitalschaltungen und mit Massenspeichern für Mikrorechnersysteme. Außerdem hält er seit 1969 an der Technischen Akademie Esslingen Vorträge zur Ingenieur-Weiterbildung.
Hobbys: Fotografieren, Mineralien sammeln
ELEKTRONIK-Leser seit 1971

Bernd Pol

Externe Speicherverfahren für Mikrocomputer

Richtig nutzen läßt sich die Mikrocomputertechnik erst, wenn zu dem im System gegebenen Speicherraum ein äußerer Massenspeicher sinnvoll hinzugefügt wird. Der folgende Beitrag stellt die für diesen Bereich (ausgenommen Floppy-Disk-Speicher [2]) wichtigsten Methoden zur externen Datenspeicherung vor. Es werden mechanische (Schallplatte, Lochstreifen), optische (Balkencodes) und magnetische (Tonbandkassetten) Datenträger betrachtet. Das Schwergewicht liegt auf einer Besprechung der verbreitetsten Methoden zur Datenspeicherung auf Tonbandkassetten mit billigen Kassettenrecordern (Tarbell-, KIM- und Kansas-City-Standard). Außerdem werden die wichtigsten Lochstreifencodes, digitale Bandaufzeichnungsverfahren, sowie Prinzipien und Lösungsvorschläge zur Datenformatierung bei der Informationsspeicherung auf Tonbandkassetten vorgestellt.

1 Problemstellung

Grundaufgabe eines jeden Rechnersystems ist die Entgegennahme, Verarbeitung und Speicherung von Daten. Dabei hängen Entgegennahme und Verarbeitung in erster Linie von der dem System gestellten Aufgabe ab. Die Datenspeicherung jedoch wird weitgehend von seiner physischen Struktur bestimmt. Prinzipiell reicht der interne Speicherraum (Arbeitsspeicher), auf dessen Inhalt unmittelbar (*random*) zugegriffen werden kann, für die Fülle der verschiedenen Daten (wobei auch Programme als Daten zu betrachten sind), die verarbeitet werden könnten, nicht aus. Es ist daher notwendig, Speicherraum außerhalb des eigentlichen Systems zu schaffen (Massenspeicher), wo die gerade nicht benötigten Daten über längere Zeiträume aufbewahrt werden können (*Bild 1*).

Bei Mikrocomputern erhält dieses Problem eine (durch den Stand der Technik bedingte) zusätzliche Dimension. Der unmittelbar erfaßbare Speicherraum ist mit in der Regel 64 KByte wesentlich kleiner als der von Großcomputern (Megabytebereich). Zudem verlieren die verwendeten Speicherbausteine üblicherweise bei Ausfall der Versorgungsspannung ihre Information. So wird bereits auf niedrigster Anwendungsebene die Bereitstellung externer Datenspeicher zu einer systembedingten Notwendigkeit. Andererseits müssen sich derartige Speichersysteme in vernünftigen Kostenrelationen zum eigentlichen Mikrocomputer halten, was den Kreis der anwendbaren Methoden beträchtlich einengt.

2 Prinzipien externer Datenspeicherung

2.1 Forderungen

Externe Datenspeicherung bedeutet Festhalten von Information in dauerhafter Form auf geeigneten physischen Trägern, die vom Rechnersystem unabhängig sind. Für den betrachteten Bereich sind hierzu folgende Forderungen zu stellen:
- preiswerter Datenträger,
- Datenträger einfach zu handhaben,
- Aufzeichnung ohne großen Aufwand zu realisieren,
- geringer Raumbedarf der Aufzeichnungs- und Leseapparatur,
- vertretbare Fehlerrate.

Der letzte Punkt hängt wechselseitig von den vorhergehenden ab. Das hat insbesondere die Konsequenz, daß die Ansprüche an die Fehlerrate nicht zu hoch getrieben werden können, soll die Speicherperipherie noch in einer vernünftigen Relation zum Aufwand für den Computer selbst stehen. Das schließt die Forderung nach möglichst hoher Schreib/Lesegeschwindigkeit ein, die zudem in hohem Maße systembedingt ist.

Die wichtigsten der derzeit üblichen Speicherverfahren sind in der *Tabelle* aufgelistet. Dabei ist jeweils angegeben, ob sich das Verfahren für den Mikrocomputerbereich eignet oder nicht.

2.2 Prinzipien

Die Datenspeicherung selbst erfolgt prinzipiell in physikalisch reproduzierbarer Form (andere, chemische oder biologische Methoden sind denkbar, aber über Spekulation nicht hinausgediehen). Nach der Art

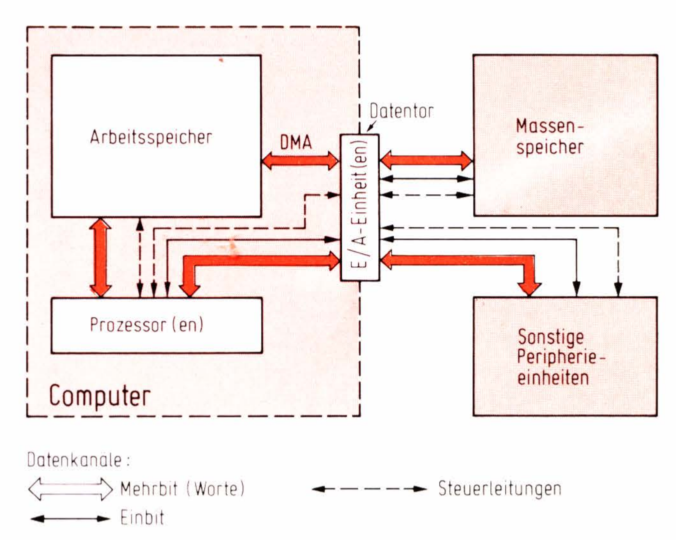


Bild 1. Grundstruktur eines Computersystems

der Datenaufzeichnung lassen sich die Verfahren unter verschiedenen Gesichtspunkten gliedern:

a) *Speicherdimension*, d. h. räumlich oder zeitlich strukturierte Aufzeichnung,
b) *Änderbarkeit* der aufgezeichneten Daten (Nur-Lese- oder Schreib/Lese-Speicher).

2.2.1 Speicherdimension

Räumliche (parallele) Aufzeichnung der Daten bedeutet unterschiedliche Markierung bestimmter Raumstellen auf oder in dem Speichermedium, beispielsweise in Matrixform. Das bietet vor allem den Vorteil, daß die Daten rasch verfügbar sind (z. B. durch Auswechseln der Speicherkarte). Nachteilig ist jedoch, daß Raumbedarf und Kosten insgesamt doch recht groß sind und die Speichermedien aufgrund ihrer Komplexität besonders sorgsam behandelt werden müssen. Außerdem ist die Austauschbarkeit der Daten zwischen verschiedenen Anwendern eingeschränkt.

Zeitliche (serielle) Aufzeichnungsmethoden umgehen diese Nachteile. Bei ihnen werden die Daten wiedererkennbar auf einem (zumeist billigen) physisch bewegten Träger in vereinbarten Zeitabständen aufgebracht. D. h. die zu speichernde Information wird zunächst aus einer räumlichen Dimension (Speichermatrix im Arbeitsspeicher) in eine zeitliche (Ausgabe als Bit- oder Bytefolge) und dann auf dem Trägermaterial wieder in eine räumliche Dimension gebracht. Das Einlesen der Daten geschieht umgekehrt. Dabei werden die gespeicherten Daten vom Computer in der Regel nicht unmittelbar verwendet, sondern zunächst in einem Pufferspeicher im Arbeitsspeicherbereich ab-

Tabelle. Methoden zur dauerhaften Datenspeicherung

Dimension	Träger	Daten änderbar	µC-Bereich	Bemerkungen
räumlich	Kernspeichermatrizen	ja	selten	teuer, nur kleine und mittlere Datenmengen, großer Raumbedarf
	Halbleitermatrizen ROM	nein	ja	Standardfestwertspeicher für
	PROM	i.d.R. nein		kleine und mittlere Datenmengen
	EPROM	bedingt		
räumlich/zeitlich (Bit-par./Wort-ser.)	Lochkarten	nein	nein	billig, langsam, einfach zu handhaben, Auswertegeräte teuer, nur kleine Datenmengen pro Karte
	Lochstreifen	nein	ja	billig, langsam, umständlicher als Lochkarten, Auswertegeräte relativ billig, mittl. Datenmengen
	Standardmagnetband	ja	nein	teuer, Standardspeicher für große Datenmengen bei Großsystemen
zeitlich	Schallplatte	nein	ja	billig, relativ langsam, kleine bis mittlere Datenmengen in Großauflage
	Balkencodes (optisch)	nein	ja	billig, einfache Vervielfältigung kleine bis mittlere Datenmengen
	Magnetplatten: feste Platten	ja	selten	teuer, schnell, sehr empfindlich Standardpufferspeicher für mittlere bis große Datenmengen
	flexible Platten (Floppy Disks)	ja	ja	relativ teuer, langsamer als feste Plattenspeicher, mittlere bis große Datenmengen, Standardmassenspeicher in µC-Systemen für raschen Zugriff
	Bandcassetten *(Compact-Kassetten)*: digitale Aufzeichnung	ja	ja	Preisklasse wie Floppy Disk, nicht so empfindlich, langsamer als sie, hohe Ansprüche an Bandgerät und -material
	Aufzeichnung mit Tonfrequenz	ja	ja	billig, recht langsam, kleine bis mittlere Datenmengen, verbreitetster Massenspeicher im Hobbybereich

gelegt (*Bild 2*). Dieser Puffer kann identisch mit dem endgültigen Speicherplatz für die Daten sein, muß es aber nicht.

Die Tatsache, daß die Daten trotz der prinzipiellen zeitlichen Dimension räumlich faßbar extern vorliegen, ermöglicht in manchen Fällen eine einfache Vervielfältigung (Schallplatte, Balkencode). Andererseits werden besondere (meist mechanische) Hilfseinrichtungen zur Datenhandhabung benötigt.

Hauptnachteil des Verfahrens ist jedoch die in der Regel recht hohe Zugriffszeit auf die gespeicherten Daten. Diese wird durch geeignete Formatierung und den oben erwähnten Pufferspeicher zwar kompensiert, bedingt dadurch aber wieder einen erhöhten Aufwand.

Die Zugriffszeit kann verringert werden, wenn die Information nicht rein seriell, d. h. Bit für Bit aufgezeichnet wird, sondern indem man je 3...8 Bit (je nach Verfahren) zu größeren Einheiten zusammenfaßt und diese einzelnen Datenworte parallel auf den Träger bringt. Dazu wird dieser in einzelne Aufzeichnungsspuren zerlegt, von denen jede einer Bitposition im Wort zugeordnet ist. Oft sind diese Worte Codes für festgelegte Informationseinheiten (Buchstaben, Ziffern, Zeichen; s.u.). Durch dieses Verfahren erhöhen sich jedoch die Kosten infolge der gestiegenen Komplexität der Aufzeichnungsapparatur und der höheren Anforderungen an das Trägermaterial.

2.2.2 Änderbarkeit

Die Aufzeichnung kann physikalisch reversibel oder irreversibel erfolgen. Reversibel, also mit der Möglichkeit, die Daten wieder zu ändern, arbeiten alle seriellen magnetischen Verfahren. Die aufgeführten optischen und mechanischen Methoden liefern dagegen in der Regel irreversible Aufzeichnungen. Dazu gehören auch die gebräuchlichen Halbleiter-Festwertspeicher (ROM, PROM und – bedingt – EPROM).

Die Möglichkeit, die Daten jederzeit wieder ändern zu können, wird erkauft durch höhere Ansprüche an Trägermaterial und Aufzeichnungsapparatur, ferner wächst (z. B. durch unabsichtliches Löschen) die Datenunsicherheit.

Irreversible Aufzeichnungen sind in der Regel billiger und da angebracht, wo die Information nicht mehr geändert zu werden braucht (z. B. bei ausgetesteten Programmen). Unter Umständen wird man aber auch (wie bei Lochstreifen) die geringen Kosten des Trägermaterials nutzen und die Daten – evtl. in größeren Abständen – jedesmal neu aufzeichnen.

2.3 *Datenträger für serielle Aufzeichnung*

Im folgenden werden nur noch die wesentlichsten seriellen Aufzeichnungsverfahren betrachtet werden. Hinweise zu Halbleiterspeichern finden sich z. B. in [1]. Bei den seriellen Verfahren soll auch auf die sogenannten Floppy-Disks nicht weiter eingegangen werden. Nähere Details dazu finden sich in [2].

Die Auswahl der Datenträger richtet sich nach dem Aufzeichnungsverfahren. Wesentlichste Forderungen an die Träger sind:

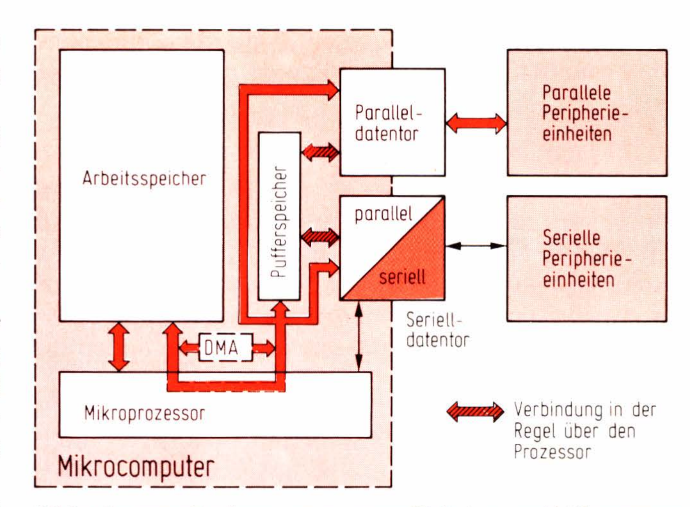


Bild 2. Datenverbindung von externen Einheiten und Mikrocomputern

– Dauerhaftigkeit des Materials,
– Stabilität der Aufzeichnung,
– leichte Handhabung.

2.3.1 Mechanische Verfahren

Zwei verschiedene Verfahren sind hier wesentlich: Lochstreifen und Schallplatte. Lochstreifen werden mechanisch oder optisch abgetastet. Sie müssen entsprechend mechanisch widerstandsfähig sein und dürfen keine zu große Transparenz besitzen. Verbreitet sind geöltes und ungeöltes Papier sowie eingefärbte Spezialkunststoffe als Trägermaterial. Lochstreifen sind auf Rollen oder im Zickzack gefaltet (*fan fold*) im Handel. Die letztere Form ist preiswerter, einfacher zu handhaben und (wegen der geringeren Masse) in Hochgeschwindigkeitslesern schneller zu verarbeiten, dafür aber auch empfindlicher gegen Schmutz und mechanische Einwirkung.

Schallplattenherstellung beginnt sich zur Verbreitung von kurzen Programmen vor allem im Hobbybereich durchzusetzen [15]. Die Daten werden hierbei (oft nach einem der später besprochenen Tonbandkassettenverfahren) im Tonfrequenzbereich codiert und im üblichen Plattenherstellungsprozeß gespeichert. Die Methode eignet sich so vor allem zur Massenverbreitung von Standardprogrammen. Wichtig ist, daß das Trägermaterial möglichst große Reinheit besitzt und der Prägeprozeß sorgfältig kontrolliert wird, um das Rauschen gering zu halten und Aussetzfehler zu vermeiden. Letztere sind der Hauptnachteil der Methode: ein Kratzer führt bereits zum Datenverlust. Man kompensiert das dadurch, daß man die Information mehrfach auf einer Platte aufzeichnet.

2.3.2 Optische Verfahren

Optische Verfahren eignen sich – soweit automatisierbar – derzeit nur zur irreversiblen Datenaufzeichnung. Sie teilen mit den Schallplatten den Vorzug leichter Reproduzierbarkeit. Im betrachteten Zusammenhang ist nur der sogenannte *Balkencode (bar code)* wichtig, bei dem die Daten in kurzen Balken verschiedener Breite verschlüsselt werden. Die Anforderungen an das Trägermaterial sind relativ gering. Es ist nur auf möglichst guten Kontrast zwischen Pa-

pier und Druckfarbe zu achten. Außerdem verbessert einigermaßen formstabiles Papier die Datensicherheit.

2.3.3 Magnetische Verfahren

Tonbandkassetten und Floppy-Disk-Speicher haben außer Lochstreifen als externes Speichermedium im Mikrocomputerbereich die größte Bedeutung. Überhaupt gehört die serielle Datenaufzeichnung unter Ausnutzen der Remanenz magnetisierbarer Materialien zu den ältesten Massenspeichermethoden der Computertechnik.

Als *Trägermaterial* dienen mit Eisenoxid- oder Chromdioxidpartikeln beschichtete Kunststoffolien, die dem Verwendungszweck entsprechend konfektioniert werden. Dabei wird gefordert:

- Homogenität der Beschichtung (Aussetzfehler),
- Abriebfestigkeit der Beschichtung (Kopfkontakt),
- Kopierfestigkeit (insbesondere Kassetten),
- Maßhaltigkeit (insbesondere Floppy Disk),
- Gleitfähigkeit (insbesondere Floppy Disk).

Die Homogenität der Beschichtung ist wesentlich für die erreichbare Aufzeichnungsdichte. Da bei digitaler Aufzeichnung zudem mit voller Durchmagnetisierung der Schicht gearbeitet wird, sind die Anforderungen an die Kopierfestigkeit recht hoch. Für rein digitale Aufzeichnung werden daher ausgesuchte (*certified*) Kassetten, oft mit Rückseitenbeschichtung, verwendet.

Zur Datenaufzeichnung mit normalen Kassettenrecordern im Tonfrequenzbereich genügen in der Regel Standardkassetten. Aber auch dort sollte man das qualitativ höchstwertige Material verwenden. Billigkassetten sind wegen der Aussetzfehler und dem viel zu hohen Bandrauschen oft unbrauchbar.

Die besonderen mechanischen Anforderungen an Floppy Disks werden durch Verwendung von festen Mylarfolien für das Trägermaterial und speziell ausgeführten (gerippten) Teflonfolien für die Auskleidung der Plattenhülle (vgl. [2a]) erfüllt.

2.4 *Übertragungsraten*

Die Geschwindigkeit serieller Datenübertragung wird in Bits pro Sekunde (bps) bzw. Baud (1 Bd ≙ 1 bps) angegeben (*Übertragungsrate*). Die erzielbare Übertragungsrate hängt von der Übertragungsmethode ab. Es sind asynchrone und synchrone Methoden zu unterscheiden (Genaueres findet sich z. B. in [5], [10], [12], [27] und [28]).

2.4.1 Asynchrone Datenübertragung

Asynchron werden serielle Daten übertragen, wenn die Taktsignale von Sender und Empfänger nicht identisch sind. Es müssen dann folgende Maßnahmen ergriffen werden (*Bild 3*):

- Vereinbarung einer bestimmten Übertragungsrate;
- Unterteilen des Datenstroms in kleinere Blöcke, für die noch ausreichende Synchronisation gewährleistet ist. Die Blocklänge wird fest vereinbart.
- Markierung von Anfang (Start) und Ende (Stopp) jedes Blocks durch zusätzliche Bits von festgelegtem logischen und zeitlichem Wert (Start- und Stoppbits).

Das Startbit hat in der Regel den logischen Pegel Null und bezeichnet den Beginn der Übertragung. Ein oder mehrere Stoppbits mit entgegengesetztem logischen Pegel dienen dann dazu, etwaige Synchronisationsfehler auszugleichen.

Mit modernen Peripheriebausteinen (z. B. Intel 8251 [10]) sind so Übertragungsraten bis zu 9600 Bd zu erreichen. Als *Standardgeschwindigkeiten* sind

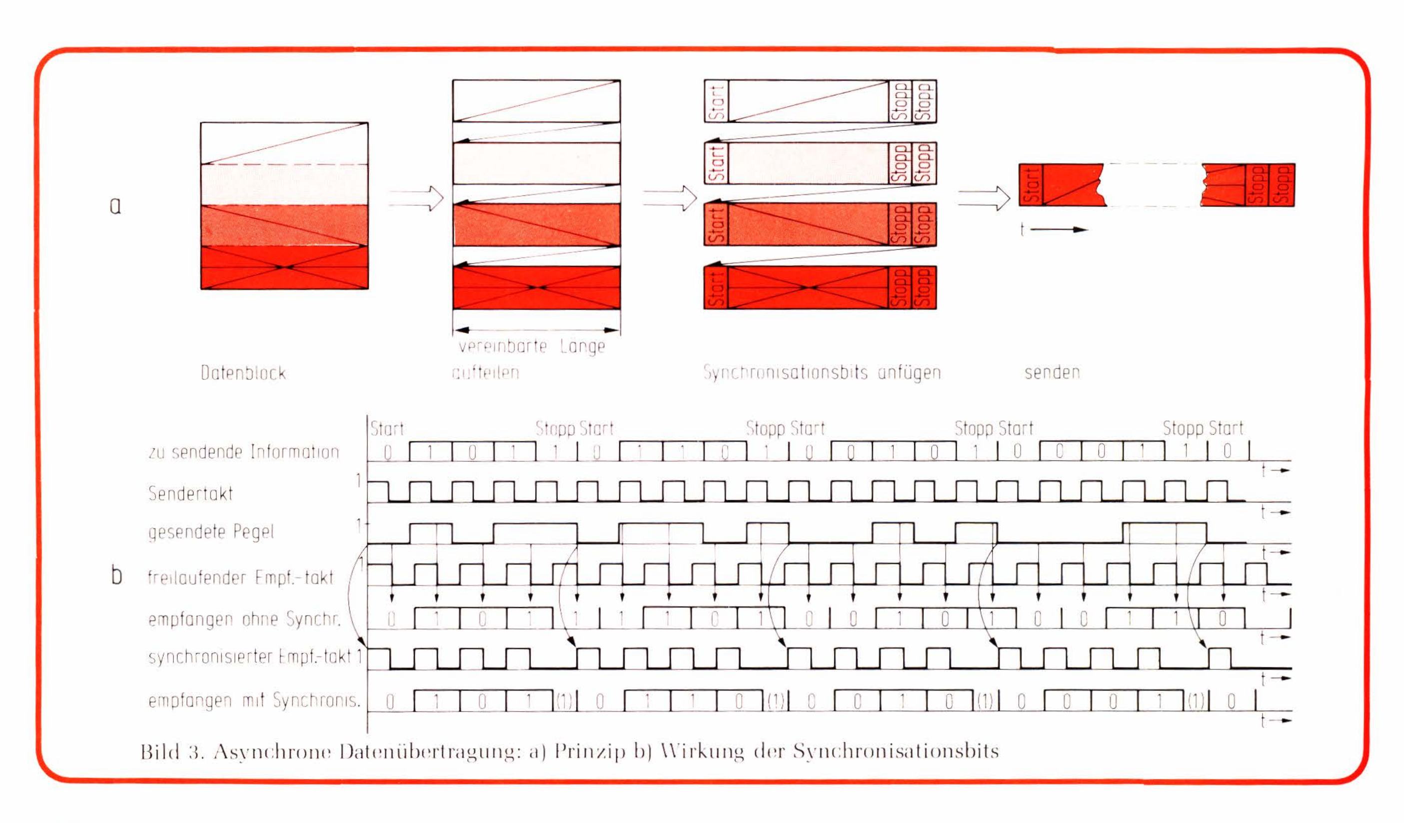


Bild 3. Asynchrone Datenübertragung: a) Prinzip b) Wirkung der Synchronisationsbits

45,45 (Telex), 110 (ASCII), 150, 300 (Kansas-City-Standard), 600, 1200, 2400, 4800, 9600 und (in Ausnahmefällen) 19200 Bd vereinbart. Dabei zählen Start- und Stoppbits mit.

2.4.2 Synchrone Datenübertragung

Hierbei arbeiten Sender und Empfänger mit demselben Taktsignal. Dieses kann sowohl von der sendenden als auch von der empfangenden Einheit zur Verfügung gestellt werden und wird auf geeignete Weise zusammen mit dem Datenstrom übertragen. Das kann durch eine besondere Datenleitung oder durch geeignetes Kombinieren von Takt- und Datensignalen geschehen (*Bild 4*). Bei synchronen Datenaufzeichnungsmethoden muß daher immer auch der Takt unmittelbar oder verschlüsselt mitgespeichert werden. Dabei sind am Anfang des Datenstroms in der Regel ein oder mehrere Markierungsbits voranzustellen, die eine saubere Einsynchronisation auf die empfangene Information ermöglichen.

Da der Datenstrom über längere Zeit ununterbrochen läuft, sind erheblich höhere Datenübertragungsraten als beim asynchronen Betrieb möglich. Der erwähnte Baustein von Intel verarbeitet bis zu 56 kBd. Das Prinzip ist jedoch sehr störanfällig gegen Laufzeit- und Phasenfehler. Daher werden die Daten in der Regel in größere Blöcke (meist je 256 bit) unterteilt, denen ein Prüfwort beigegeben wird, das die Kontrolle der Übertragung ermöglicht. Jeder Block wird (in festgelegtem Rahmen) so oft übertragen, bis er ordnungsgemäß übernommen ist. Das bedingt noch eine zusätzliche Meldeleitung vom Empfänger zum Sender zur Bestätigung der korrekten Datenübernahme (z. B. bei Floppy-Disk-Speichern, [2]).

3 Codes bei nichtmagnetischer Datenaufzeichnung

Die nichtmagnetischen Techniken zur Speicherung von Daten haben vor allem Bedeutung bei der Verteilung von Information (Programme, Standarddaten usw.). Dazu werden die Daten in verschiedenen *Codes* verschlüsselt, von denen die wichtigsten hier vorgestellt werden sollen.

3.1 Mechanische Verfahren

Lochstreifen werden in zwei Standardformaten verwendet (*Bild 5*). Sie benutzen parallel/serielle Aufzeichnung mit 5 oder 8 Kanälen. Ursprünglich als Hilfsmittel im Telegrafendienst gedacht, sind Lochstreifenstanzer und -leser oft Standardzubehör der noch häufig als Mikrocomputer-Ein/Ausgabestation verwendeten Fernschreiber. Die Aufzeichnungsmethode ist so relativ billig, aber den Standardübertragungsraten von 45,45 bzw. 110 Bd gemäß recht langsam. Schnelle Stanz- und Lesegeräte sind verfügbar. Die letzteren arbeiten meist optisch (jedem Kanal ist im Lesekopf eine Lichtschranke zugeordnet; der Bandtransport geschieht oft mit Schrittmotoren) und erreichen im Synchronbetrieb (d. h. mit Übertragen der Taktspur) Lesegeschwindigkeiten bis zu 10 kBd. Es sind Einfachstausführungen erhältlich, bei denen der Streifen von Hand durchgezogen wird. Dies hat zwar eine wesentlich höhere Beanspruchung des Trägermaterials zur Folge, mit geeigneten E/A-Schnittstellen erreicht man aber (als Spitzenwert) die angegebene Übertragungsrate.

3.1.1 Standardcodierungen

5-Kanal-Lochstreifen werden üblicherweise im CCIT-Codieralphabet Nr. 2 („Baudot"-Code [eigent-

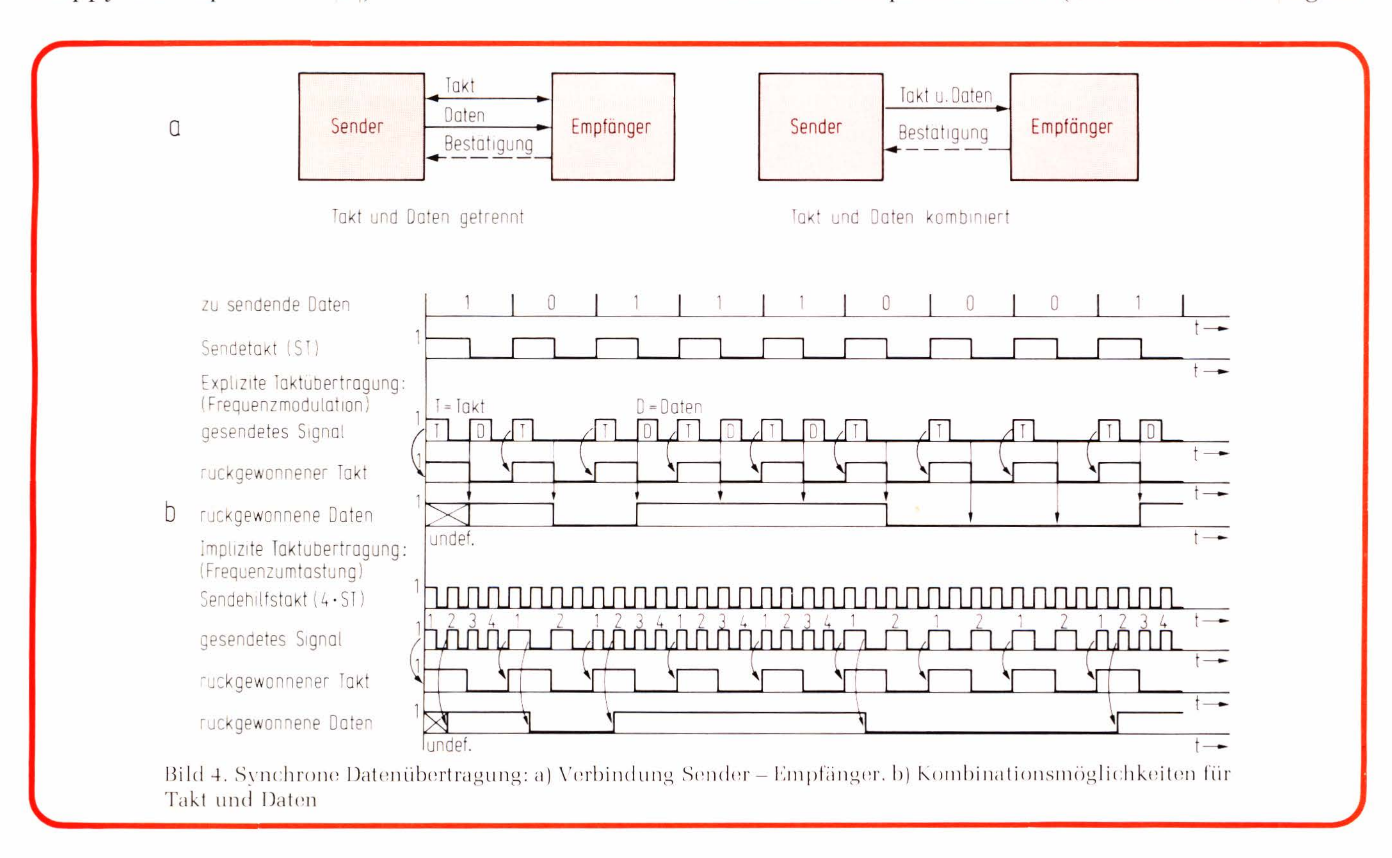


Bild 4. Synchrone Datenübertragung: a) Verbindung Sender – Empfänger, b) Kombinationsmöglichkeiten für Takt und Daten

lich Murray-Code) verschlüsselt, 8-Kanal-Lochstreifen im ISO-7-bit-Code (ASCII-Code, ASCII = *American Standard of Information Interchange*) (*Bild 6a, b*). Der freie achte Kanal dient hier meist als Paritätsinformation oder bleibt ganz leer. Beide Paritätsarten, gerade und ungerade, sind zum Test auf Übertragungsfehler üblich. Man nutzt die Tatsache aus, daß durch Fehler in einem Bit pro Wort die Parität gerade umgekehrt wird.

Bevor der ASCII-Code international genormt wurde, war der Firmenstandard EBCDIC (*Extended Binary-Coded-Dezimal Information Code*) von IBM weit verbreitet. Dieser 8-bit-Code wird noch heute bei Lochkarten in größerem Maßstab verwendet. Seiner Bedeutung wegen ist er in *Bild 6c* wiedergegeben. Es handelt sich um eine Erweiterung des bekannten BCD-Codes von vier auf acht Bit. Von den so möglichen 256 Zeichen sind in der derzeitigen Fassung 152 in 8 Gruppen codiert. Eine etwas abgewandelte Form, der sogenannte EBC-Correspondence Code, kommt mit nur 6 bit aus und wird zur Ansteuerung der elektrischen Schreibmaschinen von IBM verwendet.

3.1.2 Weitere wichtige Codes

Vor allem zum Datenverkehr zwischen Benutzer und Herstellerfirma, aber auch zum Datenaustausch größerer Benutzergruppen sind weitere drei (vier) Codes von Bedeutung. Sie dienen vor allem der Programmweitergabe und als Unterlage zur Programmierung von Festwertspeichern. Es handelt sich um parallel/serielle oktale und sedezimale (hexadezimale) Codes sowie um einen rein seriellen Code über einem Vorrat von vier Zeichen.

Die beiden oktalen Formate RIM (*Read In Mode*) und BIN (*BINary*) (*Bild 7a, b*) wurden von der Digital Equipment Corporation für ihre verbreitete PDP-8-Minicomputer-Serie entwickelt. Sie haben durch den CMOS-Prozessor IM 6100 von Intersil auch Bedeutung im Mikrocomputerbereich gewonnen, da dieser Prozessor den PDP-8e-Befehlssatz benutzt ([7], [8], [9])

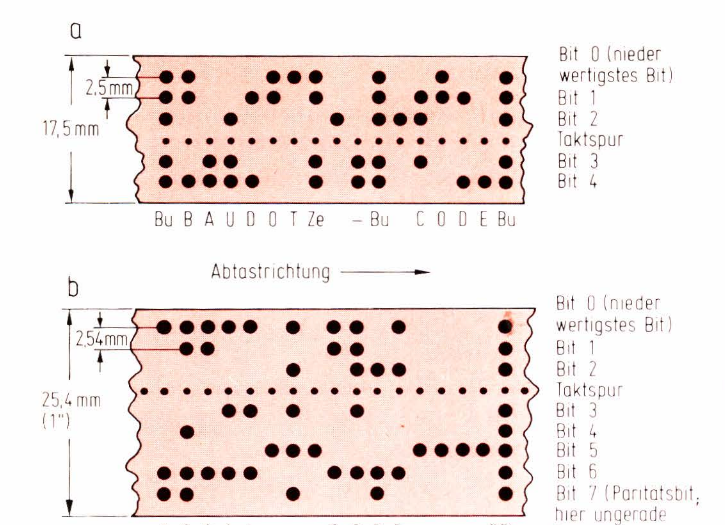


Bild 5. Lochstreifen: a) 5-Kanal, b) 8-Kanal

a

Bit 3 / 4 → ; 2 1 Q ↓	0 / 0 B	Z	1 / 0 B	Z	0 / 1 B	Z	1 / 1 B	Z
0 0 0			≡	≡	E	3	A	–
0 0 1	T	5	L	)	Z	+	W	2
0 1 0	<	<	R	4	D	✠	J	Ω
0 1 1	0	9	G		B	?	Ze	Ze
1 0 0	ᴗ	ᴗ	I	8	S	'	U	7
1 0 1	H		P	0	Y	6	Q	1
1 1 0	N	,	C	:	F		K	C
1 1 1	M	.	V	=	X	/	Bu	Bu

< = Wagenrücklauf
ᴗ = Leerschritt
≡ = Zeilenvorschub
✠ = Anrufzeichen (Wer da ?)
Ω = Glocke
Ze = Walze auf Zeichen (Z) umschalten
Bu = Walze auf Buchstaben (B) umschalten

b

Bit 4 5 6 → ; 3 2 1 0 ↓	0 0 0	CNTRL	1 0 0	CNTRL	0 1 0	1 1 0	0 0 1	1 0 1	0 1 1	1 1 1
0 0 0 0	NUL	@	DLE	P	SP	0	@	P	`	p
0 0 0 1	SOH	A	DC 1 X-ON	Q	!	1	A	Q	a	q
0 0 1 0	STX	B	DC 2 TAPE	R	"	2	B	R	b	r
0 0 1 1	ETX	C	DC 3 X-OFF	S	#	3	C	S	c	s
0 1 0 0	EOT	D	DC 4 ~~TAPE~~	T	$	4	D	T	d	t
0 1 0 1	ENQ	E	NAK	U	%	5	E	U	e	u
0 1 1 0	ACK	F	SYN	V	&	6	F	V	f	v
0 1 1 1	BEL	G	ETB	W	´	7	G	W	g	w
1 0 0 0	BS	H	CAN	X	(	8	H	X	h	x
1 0 0 1	HT	I	EM	Y	)	9	I	Y	i	y
1 0 1 0	LF	J	SUB	Z	*	:	J	Z	j	z
1 0 1 1	VT	K	ESC	[	+	;	K	[	k	{
1 1 0 0	FF	L	FS	/	,	<	L	\	l	ǀ
1 1 0 1	CR	M	GS	]	–	=	M	]	m	} ALT MODE
1 1 1 0	SO	N	RS	^	.	>	N	^ ↑	n	~
1 1 1 1	SI	O	US	–	/	?	O	_ ←	o	DEL RUB OUT

Wiedergegeben in der üblichen englischen Bezeichnungsweise.
Die Spalte CNTRL bezieht sich auf die oft angegebene Lage der Steuerzeichen bei Umschaltung mit der CONTROL-Taste. Mehrfachangaben in einem Feld sind alternative Bezeichnungsweisen.

c

Bit 4 5 6 7 → ; 3 2 1 0 ↓	0 0 0 0	1 0 0 0	0 1 0 0	1 1 0 0	0 0 1 0	1 0 1 0	0 1 1 0	1 1 1 0	0 0 0 1	1 0 0 1	0 1 0 1	1 1 0 1	0 0 1 1	1 0 1 1	0 1 1 1	1 1 1 1
0 0 0 0	NUL	DLE	DS		SP	&	-						{	}	\	0
0 0 0 1	SOH	DC1	SOS				/		a	j	~		A	J		1
0 0 1 0	STX	DC2	FS	SYN					b	k	s		B	K	S	2
0 0 1 1	ETX	TM							c	l	t		C	L	T	3
0 1 0 0	PF	RES	BYP	PN					d	m	u		D	M	U	4
0 1 0 1	HT	NL	LF	RS					e	n	v		E	N	V	5
0 1 1 0	LC	BS	ETB	UC					f	o	w		F	O	W	6
0 1 1 1	DEL	IL	ESC	EOT					g	p	x		G	P	X	7
1 0 0 0		CAN							h	q	y		H	Q	Y	8
1 0 0 1	RLF	EM						\	i	r	z		I	R	Z	9
1 0 1 0	SMM	CC	SM		¢	!	¦	:								LV
1 0 1 1	VT	CU1	CU2	CU3	.	$	,	#								
1 1 0 0	FF	IFS		DC4	<	*	%	@					⑀		⑇	
1 1 0 1	CR	IGS	ENQ	NAK	(	)	–	'								
1 1 1 0	SO	IRS	ACK		+	;	>	=					⑂			
1 1 1 1	SI	IUS	BEL	SUB	ǀ	¬	?	"								E

Bild 6. Standardcodes: a) Baudot-Code, b) ASCII-Code, c) EBCDIC-Code [32]

und deshalb auf die umfangreiche Programmbibliothek von DEC zurückgreifen kann.

Das vorgestellte sedezimale Format *(Bild 8)* hat weite Verbreitung durch den Einkartencomputer KIM von MOS-Technology gefunden. Hier wird jedes 8-bit-Wort (Byte) in zwei 4-bit-Hälften *(Nibbles)* zerlegt, die als ASCII-Zeichen (ihrem Wert entsprechend) interpretiert und auf den Lochstreifen gestanzt werden. Das Format enthält noch Informationen über Anfang, Inhalt und Ende der Aufzeichnung.

Etwas exotisch mutet das von der Firma Intel als Unterlage zur ROM-Programmierung verwendete BNPF-Verfahren *(Bild 9)* an. Die Buchstaben stehen für Beginning (Anfang eines Bytes), Negative, Positive (Logischer Pegel der Worte nach Programmierung) und Finish (Ende des Bytes). Es handelt sich also um ein asynchrones Übertragungsverfahren, das – obwohl auf 8- oder 5-Kanal Lochstreifen aufgezeichnet – rein seriell und entsprechend langsam ist. Da bei der Programmierung von Festwertspeichern die Geschwindigkeit jedoch ohnehin eine untergeordnete Rolle spielt, die Datensicherheit aber ausgesprochen wichtig ist, hat sich das Verfahren als sehr vorteilhaft erwiesen. Ein weiterer Vorteil besteht darin, daß man in den Code Bemerkungen (soweit sie nicht die Buchstaben B oder F enthalten) einfügen kann, ohne die Datenübertragung zu beeinflussen (vgl. dazu [10], S. 7–14).

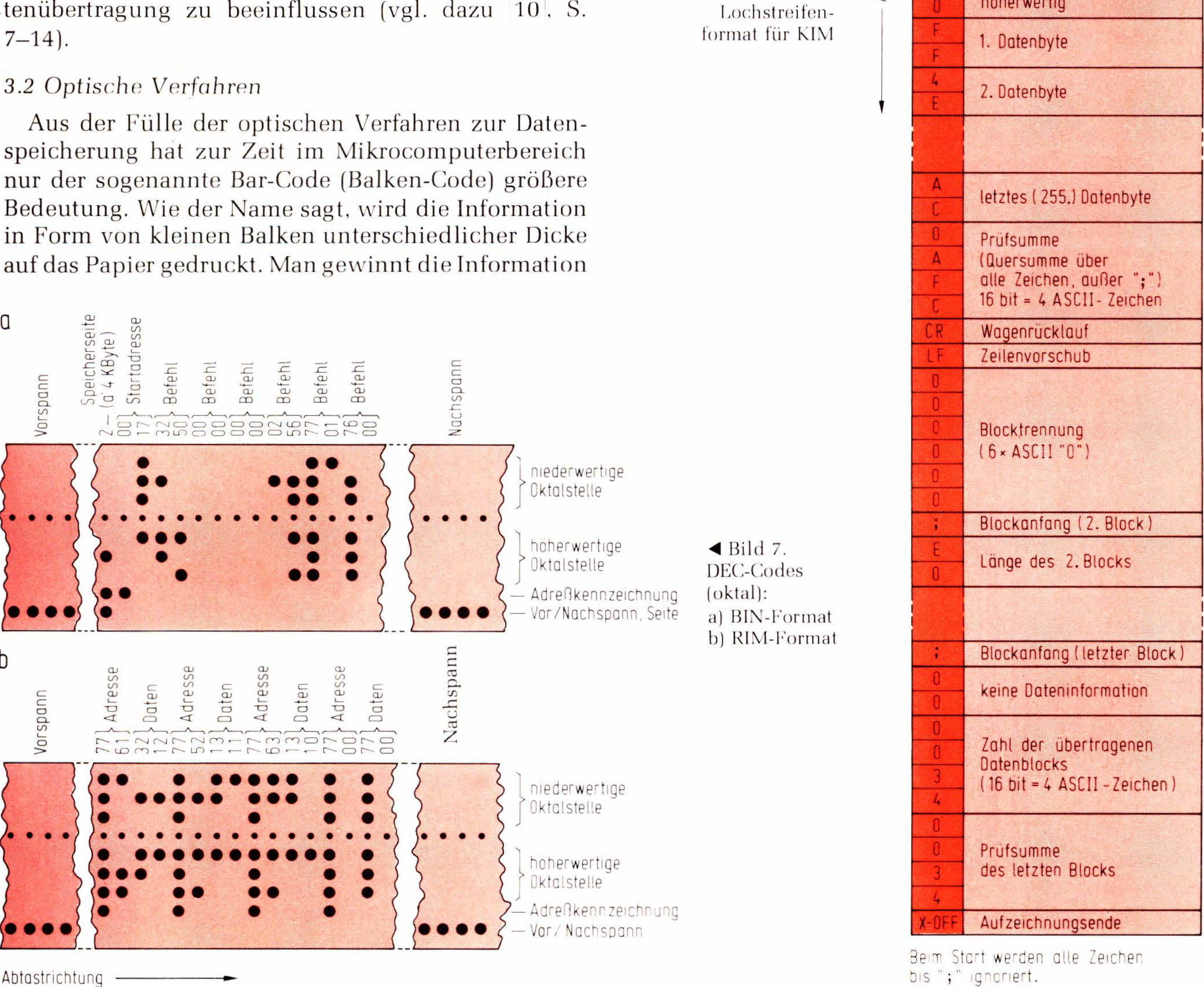
3.2 *Optische Verfahren*

Aus der Fülle der optischen Verfahren zur Datenspeicherung hat zur Zeit im Mikrocomputerbereich nur der sogenannte Bar-Code (Balken-Code) größere Bedeutung. Wie der Name sagt, wird die Information in Form von kleinen Balken unterschiedlicher Dicke auf das Papier gedruckt. Man gewinnt die Information mit einem Lichtgriffel durch Abtasten mit einer Fotozelle zurück. Die einfachere Form (der eigentliche Bar-Code) hat in den USA zur Massenverbreitung von Programmen beträchtliche Verbreitung gefunden. Eine etwas kompliziertere Form, bei der alphanumerische Zeichen als Zusatzinformation für den Benutzer verwendet werden, ist der zur Warenauszeichnung gebräuchliche UPC-Code *(Universal Product Code)*. *Bild 10* gibt Beispiele für beide Codes, näheres findet sich in [3], [4] (Bar-Code) und [5], S. 5–12...5–16 (UPC-Code).

4 Magnetische Datenaufzeichnung

4.1 *Problematik*

Die Aufzeichnung digitaler Information auf Magnetband oder -platte unterscheidet sich im Grundprinzip nicht wesentlich von der akustischer Informationen. Im Prinzip können (und werden) auch ge-



◄ Bild 7. DEC-Codes (oktal): a) BIN-Format b) RIM-Format

Bild 8. ► Lochstreifenformat für KIM

wöhnliche, für die Unterhaltungselektronik entwickelte Tonbandgeräte zur Datenspeicherung herangezogen werden. Dennoch empfiehlt es sich, digitale Daten mit speziell dafür entwickelten Verfahren und Geräten aufzuzeichnen, wenn höhere Anforderungen an Datensicherheit, Übertragungsgeschwindigkeit und Datenhandhabung gestellt werden.

Geräte zur Aufzeichnung akustischer Information sollen im wesentlichen drei Bedingungen erfüllen:

- Wiedergabe möglichst unterschiedlicher Lautstärkepegel (Dynamik),
- weiter Frequenzumfang bei gleichzeitig
- geringen harmonischen Verzerrungen (Klirrfaktor).

Diesen Forderungen wird durch Gegenkopplungsnetzwerke, automatische Verstärkungsregelung bei Aufnahme und Wiedergabe und Hf-Vormagnetisierung des Bandmaterials Rechnung getragen. Für die angestellte Betrachtung wirken sich die Gegenkopplungsnetzwerke am ungünstigsten aus. Sie erzeugen nämlich frequenzabhängige Phasenverschiebungen im Nutzsignal und im Zusammenhang mit anderen Maßnahmen unterschiedliche Gruppenlaufzeiten für einzelne Signalkomponenten. Im Unterhaltungssektor sind derartige Fehler vernachlässigbar. In der Digitaltechnik dagegen kommt es auf

- exakte Signal/Zeit-Beziehungen und
- möglichst einheitliche und weitreichende Frequenzauflösung (der Rechteckimpulse wegen) an.

Vor allem die erste Forderung wird durch die erwähnten Fehler berührt. Die Verschleifung der Rechteckimpulse durch Verletzen der zweiten Bedingung ist eher tolerabel, da sie in weitem Umfang mit relativ



Modul	linke Ziffern (ungerade Parität) 0	1	2	3	4	5	6	rechte Ziffern (gerade Parität) 0	1	2	3	4	5	6
0	0	0	0	1	1	0	1	1	1	1	0	0	1	0
1	0	0	1	1	0	0	1	1	1	0	0	1	1	0
2	0	0	1	0	0	1	1	1	1	0	1	1	0	0
3	0	1	1	1	1	0	1	1	0	0	0	0	1	0
4	0	1	0	0	0	1	1	1	0	1	1	1	0	0
5	0	1	1	0	0	0	1	1	0	0	1	1	1	0
6	0	1	0	1	1	1	1	1	0	1	0	0	0	0
7	0	1	1	1	0	1	1	1	0	0	0	1	0	0
8	0	1	1	0	1	1	1	1	0	0	1	0	0	0
9	0	0	0	1	0	1	1	1	1	1	0	1	0	0

Bild 9. Intel-BNPF-Format

Bild 10. Balkencodes: a) Bar-Code, Programmausschnitt, b) Bar-Code, Organisation einer Zeile, c) UPC-Standardsymbol, d) Konstruktion einer Ziffer im UPC-Code, e) Codierung der Ziffern im UPC-Code

einfachen Methoden aus dem Signal zurückgewonnen werden können.

Durch die Phasenfehler ergibt sich die scheinbar paradoxe Folgerung, daß z. B. Kassettenrecorder um so weniger zur Datenaufzeichnung taugen, je hochwertiger sie sind. Allerdings sind die Lauffehler bei Niedrigpreisgeräten manchmal auch nicht zu vernachlässigen. In der Praxis haben sich Geräte der unteren Mittelklasse in Verbindung mit hochwertigen Bändern am besten bewährt, falls nicht ohnehin die speziell zur Datenaufzeichnung entworfenen Geräte vorgezogen werden. Diese verfügen in der Regel noch über zusätzliche mechanische Einrichtungen, die auch die Datenhandhabung und die Speicherdichte verbessern. Dazu gehören:

- präzise Mechanik für Laufkonstanz und Start/Stopp-Steuerung;
- Lesebetrieb vor- und rückwärts;
- schneller Suchlauf nach Datenblöcken vor- und rückwärts.

Diese Eigenschaften verteuern die Geräte allerdings im Mittel um den Faktor 5. Im semiprofessionellen und Hobbybereich werden daher übliche Kassettenrecorder vorgezogen. Sie sollten folgende Forderungen erfüllen (vgl. [18], [21], [23], [25], [30]):

- Frequenzumfang bis mindestens 8000 Hz.
- Zu bevorzugen ist ein Diodenausgang, dem dann beispielsweise ein einfacher Operationsverstärker nachgeschaltet werden kann. Dadurch läßt sich ein Teil der Gegenkopplungsnetzwerke umgehen, und man behält eine (erträgliche) akustische Kontrolle über den Schreib/Lese-Vorgang.
- Andernfalls ist der Abschluß über die Kopfhörer/Zweitlautsprecheranschlüsse notwendig. Dazu sollte unbedingt ein Klangregler vorhanden sein, der auf höchste Wiedergabefrequenz gestellt werden muß (ohne Klangregler werden in der Regel die hohen Frequenzen im Gerät abgesenkt, was die Impulsantwort sehr verschlechtert). Außerdem muß der eingebaute Lautsprecher beim Betrieb abschaltbar sein, will man (wegen der notwendigen großen Lautstärkepegel) nicht binnen kurzem irrenhausreif sein.
- Ein eingebautes Mikrofon muß selbstverständlich beim Aufnahmevorgang abgeschaltet werden können.
- Automatische Aussteuerung der Aufzeichnung ist von Vorteil, aber nicht zwingend notwendig (allerdings muß dann sehr sauber ausgesteuert werden, da Übersteuerungen des Bandes zu beträchtlichen Phasen- und Laufzeitfehlern durch die Gegenkopplungsnetzwerke im Leseverstärker führen können [30]).
- Sollen pro Band mehrere Aufzeichnungen verwendet werden, so ist zur Buchführung ein Bandzähler unbedingt notwendig.
- Sehr wünschenswert ist eine automatische Start/Stopp-Steuerung. Sie kann in der Regel durch Einfügen eines Relais in die Motorleitungen nachgerüstet werden und gestattet den automatischen Bandbetrieb über das Programm.

4.2 *Digitale Aufzeichnung*

Da bei digitaler Information nur zwei Zustände, „0" und „1", zu unterscheiden sind, kann man das Band bei der Aufzeichnung voll durchmagnetisieren (geeignete Auswerteelektronik vorausgesetzt). Man ordnet den logischen Pegeln dann je eine Orientierungsrichtung der remanenten Dipole auf dem Band zu. Die Speicherdichte wird in bits per inch (bpi) Bandmaterial beschrieben. Standardspeicherdichte sind 800 bpi (ECMA-34-Standard, ECMA = *European Computer Manufacturers Association*), das entspricht etwa 31,5 bit/mm. Wird dann noch das Aufzeichnungsverfahren vereinbart, so sind bespielte Bänder leicht austauschbar (ECMA-kompatibel sind im NRZ-Verfahren phasencodiert beschriebene Bänder – auch „Manchester"-Standard genannt).

Es spielt dabei keine Rolle, mit welcher Geschwindigkeit die Daten von dem jeweiligen Gerät aktuell verarbeitet werden (man findet abhängig von der Bandgeschwindigkeit häufig Übertragungsraten von 3000, 6000 oder 12 000 Bd, jedoch werden auch Geräte mit einstellbarer Bandgeschwindigkeit angeboten).

Für die Aufzeichnung selbst sind verschiedene Verfahren üblich ([13], S. 9, 17 ff.). Sie unterscheiden sich nach der Codierung der aufzuzeichnenden Daten durch den Schreibstrom (*Bild 11*).

4.2.1 Impulsschrift (*Return to Zero*, RZ)

Hier wird jedem Bit ein Impuls des Schreibstroms zugeordnet, der kürzer als die Dauer des Bits selbst (Bitzelle) sein muß. Der logische Pegel wird durch die Richtung des Schreibstroms gegeben. Zwischen zwei Schreibimpulsen kehrt der Strom stets auf Null zurück, woher die englische Bezeichnung des Verfahrens kommt.

Zur Auswertung wird in der Regel der zweite Impuls jeder Bitzelle ausgeblendet, wodurch man für eine logische 1 stets einen positiven, für eine logische 0 stets einen negativen Impuls (beim gewählten Beispiel der Schreibstromrichtung) erhält. Da auf diese Weise der Aufzeichnungstakt mit zurückgewonnen wird, handelt es sich um ein synchrones Verfahren.

4.2.2 Dauerstromschrift (*No Return to Zero*, NRZ)

Das RZ-Verfahren hat den Nachteil, daß pro Bit zwei Flußwechsel auf dem Band nötig sind. Dadurch wird die erzielbare Aufzeichnungsdichte verringert (man zeichnet jedes Bit praktisch zweimal auf). Diese Redundanz wird umgangen, indem man den Schreibstrom bis zum Wechsel der Information beibehält. Eine Folge von logischen Einsen wird so z. B. durch positive, von logischen Nullen durch negative Stromrichtung gekennzeichnet. Der Wechsel geschieht in der Mitte der Bitzelle.

Das Verfahren gestattet eine recht hohe Speicherdichte auf dem Band, bedingt aber wegen des dauernd fließenden Schreibstroms einen beträchtlichen Leistungsaufwand. Als weiterer Nachteil kommt hinzu, daß die Taktinformation entfällt, weshalb hochpräzise Laufwerke benötigt werden.

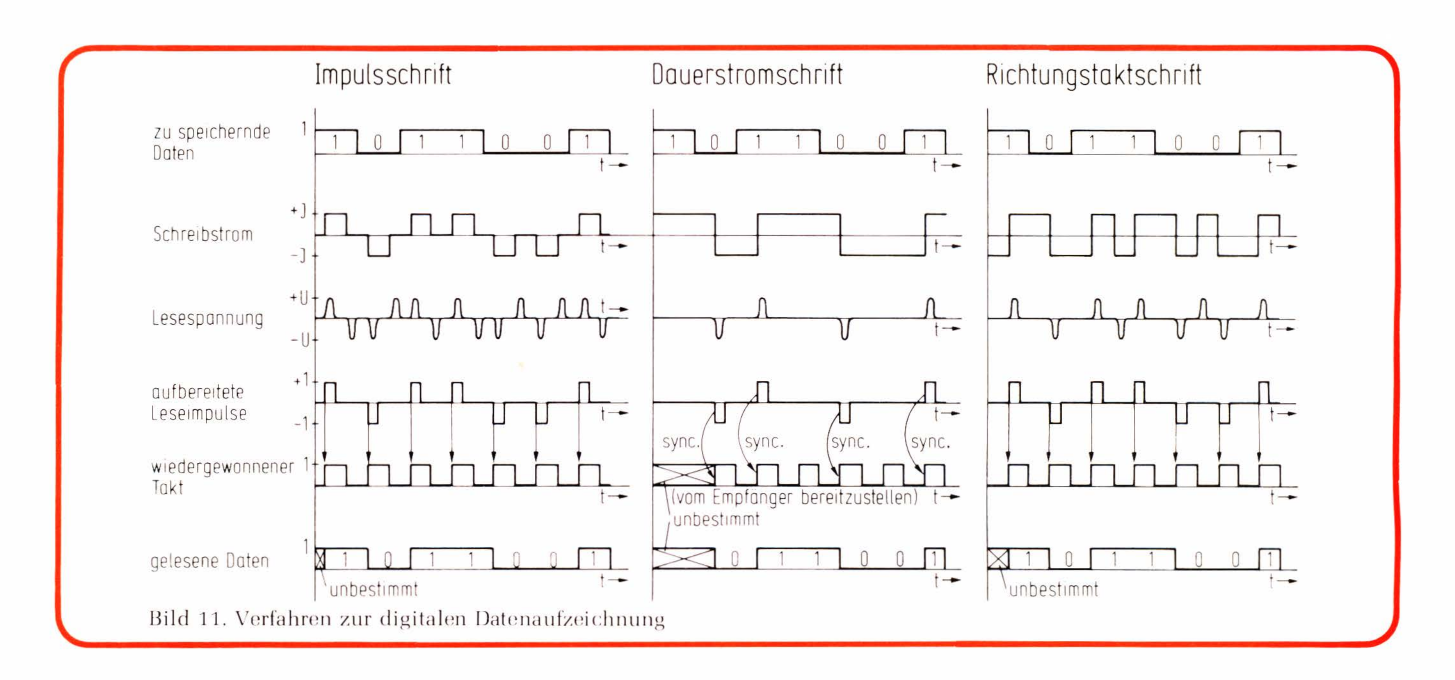


Bild 11. Verfahren zur digitalen Datenaufzeichnung

4.2.3 Richtungstaktschrift *(Phase Shift Keying*, NRZ)

Die Nachteile lassen sich umgehen, wenn man jeder Bitzelle genau einen Flußwechsel zuordnet, dergestalt, daß innerhalb der Zelle einer logischen Eins der Schreibstrom von negativer nach positiver Richtung, innerhalb einer logischen Null von positiver nach negativer Richtung wechselt. Das erfordert bei einer Serie gleicher Bits je einen zusätzlichen Richtungswechsel an den Bitzellengrenzen. Auf diese Weise ist der Schreibstrom im zeitlichen Mittel gleich Null.

Beim Auslesen des Bandes erhält man die Taktinformation auf einfache Weise zurück. Die erzielte Speicherdichte ist etwas geringer als beim reinen NRZ-Verfahren, die Auswerteelektronik in etwa der beim RZ-Verfahren gleich.

Technisch erreicht man die Aufzeichnung einfach, indem man den Aufzeichnungstakt mit der zu schreibenden Information EXKLUSIV-ODER-verknüpft *(Bild 12)*. Es resultiert so an den Grenzen verschiedenwertiger Bits ein Phasensprung von 180 Grad, woher die Bezeichnung Phasenumtastung *(Phase Shift Keying, PSK)* für dieses Verfahren rührt.

4.3 Aufzeichnung mit üblichen Kassettenrecordern

Die beschriebenen Verfahren eignen sich nur mit Einschränkungen zur Verwendung mit handelsüblichen Kassettenrecordern. Man hat hier nach Verfahren zur Datenaufzeichnung gesucht, die

- unempfindlich gegen Phasenschwankungen,
- unempfindlich gegen Geschwindigkeitsschwankungen sein sollten und mit
- kleiner Bandbreite

auskommen, den Konstruktionsbesonderheiten von im Tonfrequenzbereich arbeitenden Geräten also entsprechen.

4.3.1 Phasenumtastung: Tarbell-Interface

Eines der ersten Mikrocomputer-Interfaces, das die Datenaufzeichnung mit handelsüblichen Kassettenrecordern gestattet, wurde 1972/73 von *Donald Tarbell* entwickelt [20]. Es arbeitet nach dem NRZ-Phasenumtastverfahren mit 800 bpi *(Bild 13)* und hat in den USA weite Verbreitung gefunden. Die Methode erlaubt eine Datenübertragungsrate von ca. 1500 Bd; 256 Byte werden in 1,4 s übertragen.

Die Praxis hat gezeigt, daß dieses Verfahren bei einigen Kassettenrecordern (aus den vorher angegebenen Gründen) nicht funktioniert. Bei einigen Geräten aus der unteren Mittelklasse wurden jedoch selbst mit größeren Speicherdichten noch brauchbare Ergebnisse erzielt [26]. Das liegt zum großen Teil an der synchronen Übertragungsmethode, die Geschwindig-

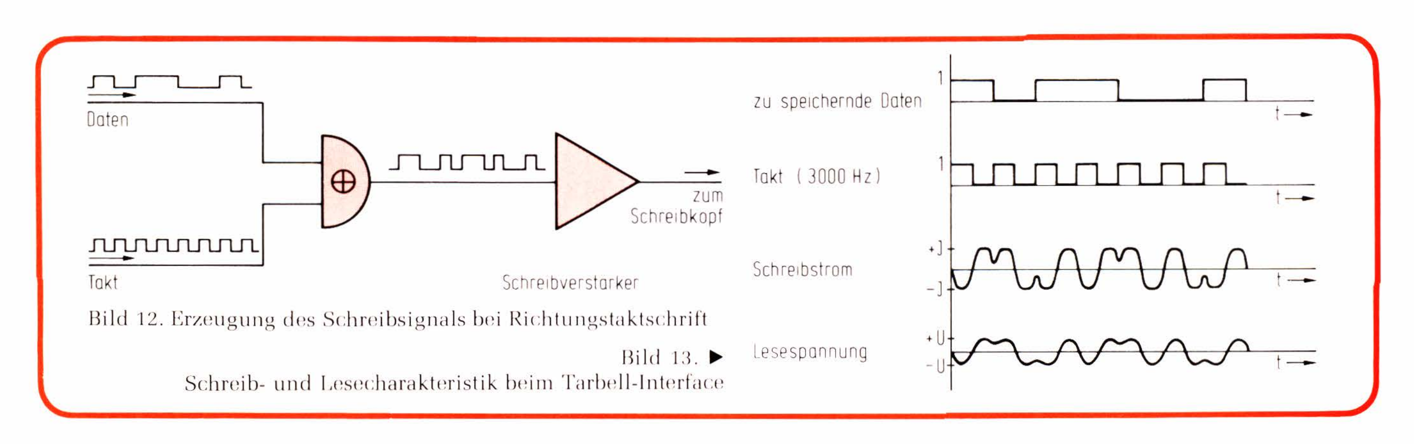


Bild 12. Erzeugung des Schreibsignals bei Richtungstaktschrift

Bild 13. ▶ Schreib- und Lesecharakteristik beim Tarbell-Interface

keits- und teilweise auch Phasenschwankungen selbsttätig korrigiert.

Ein anderes Problem bei der Weitergabe von mit dem Interface aufgezeichneten Programmen besteht darin, daß die Aufzeichnungen unter Umständen verschiedene Phasenlage haben können. Bei der gewählten hardwaremäßigen Decodierung des Signals kann es unter Umständen bei falscher Phasenlage zu Funktionsstörungen kommen [26].

4.3.2 Frequenztastung: TK-80

Ein weiterer Nachteil des Phasenumtastverfahrens liegt in der Tatsache, daß aus dem verschliffenen Signal, das vom Recorder geliefert wird, die Information nur schwierig zurückzugewinnen ist [26]. Andere Lösungen arbeiten daher nicht mit Phasen- sondern mit Frequenzcodierung der Information.

Einer der einfachsten Wege dazu wurde bei dem Einkarten-Mikrocomputer TK-80 beschritten [14]. Hier wird bei einer logischen 1 einfach ein Rechteckoszillator ein-, bei einer logischen 0 ausgeschaltet. Das Übertragungsformat ist dem Standardverfahren für 7-Kanal-Fernschreiber angepaßt und überträgt mit 110 Bd 1 Startbit, 8 ASCII-codierte Bits und 3 Stoppbits *(Bild 14)*. Das Interface kann so recht einfach gehalten werden, und zur Codierung/Decodierung genügt das ohnehin vorhandene Fernschreiber (TTY)-Interface. Geschwindigkeitsschwankungen des Bandes werden durch die drei Stoppbits ausgeglichen. Phasenfehler sind durch die niedrige Übertragungsrate im Verein mit der Frequenztastung so gut wie wirkungslos. Allerdings dauert die Übertragung eines Datenblocks von 256 Byte nahezu eine halbe Minute.

4.3.3 Bitorientierte Frequenzumtastung: KIM

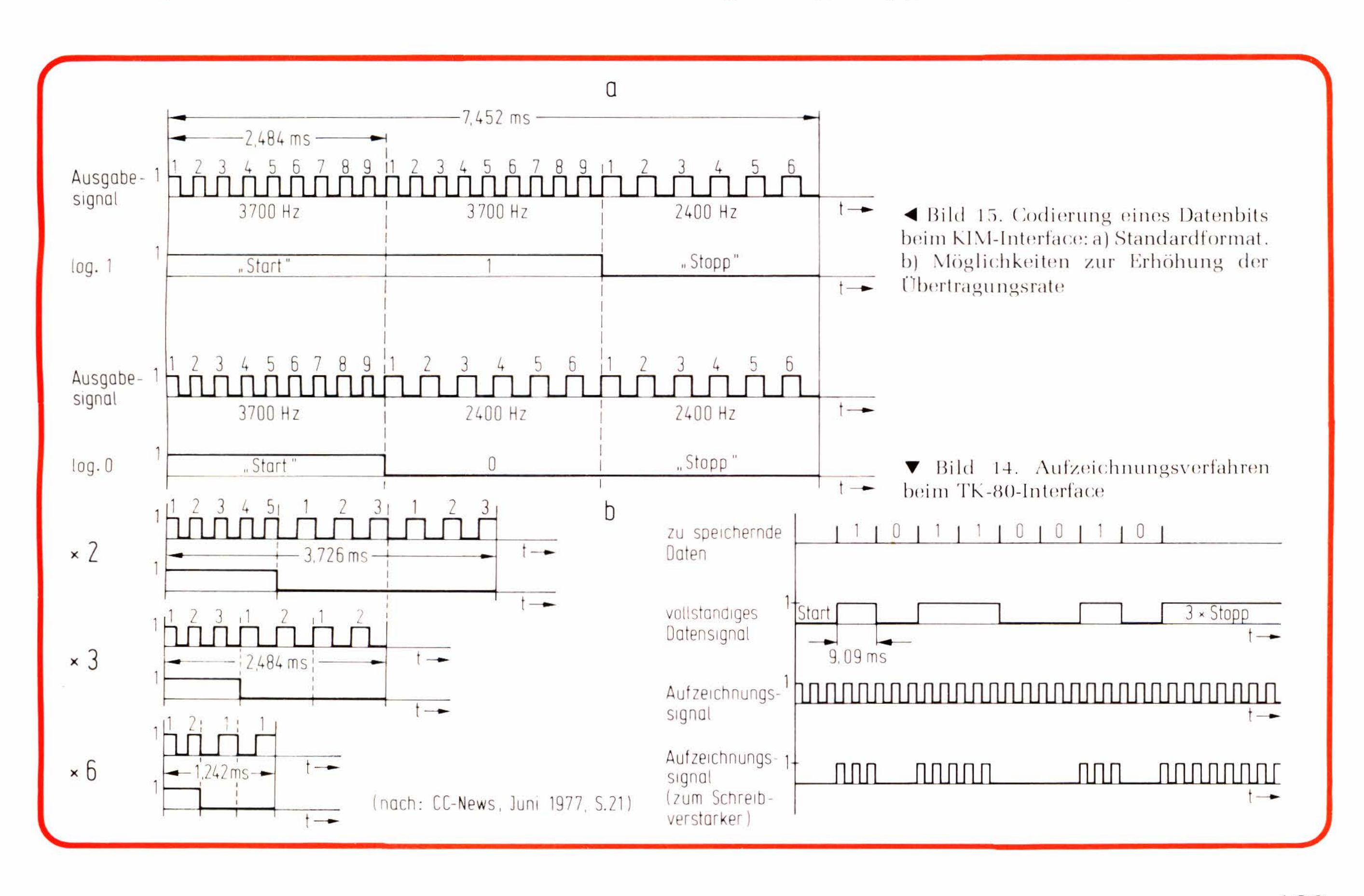
Ein üblicherer Weg zur Frequenzcodierung besteht darin, statt einer ein- und ausgeschalteten Frequenz zwei zu verwenden, zwischen denen der Information und Codierung entsprechend gewechselt wird. Das gibt eine deutlich höhere Datensicherheit.

Ein übriges zur Datensicherheit läßt sich tun, wenn man

- die Information möglichst redundant und
- die einzelnen zu übertragenden Bits möglichst unverwechselbar codiert.

Beide Wege wurden beim Kassetten-Interface des bekannten Einkarten-Mikrocomputers KIM beschritten. Hier wird jedes einzelne Bit im Frequenzumtastverfahren *(Frequency Shift Keying, FSK)* verschlüsselt *(Bild 15 a)*. Man verwendet die beiden „Eckfrequenzen" 3700 Hz und 2400 Hz. Jede Bitzelle wird in drei Teile unterteilt. Das erste Drittel enthält immer 9 Schwingungen zu 3700 Hz, das letzte immer 6 Schwingungen zu 2400 Hz. Als eigentlicher Informationsträger dient das mittlere Bitzellendrittel: log. 0 ist hier durch 3700 Hz, log. 1 durch 2400 Hz codiert. Insgesamt erhält eine Bitzelle so die fixierte Länge von 7,452 ms.

Jedes Bit ist durch den Sprung von 2400 Hz auf 3700 Hz am Bitzellenanfang eindeutig markiert. Daraus läßt sich die Taktinformation zurückgewinnen. Von daher läßt sich das Verfahren als synchron beschreiben. Durch die eindeutige Kennzeichnung jedes Bits läßt es sich aber auch asynchron verwenden. Ein wesentliches Merkmal der Methode ist die nahezu vollständige Festigkeit gegenüber Fremdfrequenzen, was das



◄ Bild 15. Codierung eines Datenbits beim KIM-Interface: a) Standardformat, b) Möglichkeiten zur Erhöhung der Übertragungsrate

▼ Bild 14. Aufzeichnungsverfahren beim TK-80-Interface

Einfügen gesprochener Kommentare in die Aufnahme ohne Störung des Übertragungsverfahrens erlaubt.

Nominell ergibt sich eine Übertragungsrate von 134 Bd: 256 bit könnten in 16 s übertragen werden. Wegen des für die Datenblöcke gewählten Formats jedoch dauert es in der Praxis etwa doppelt so lange. Es gibt daher eine Reihe von Alternativverfahren, die mit Informationskompression oder Umdefinieren der Bitcodierung arbeiten und eine Versechsfachung der Übertragungsrate ermöglichen *(Bild 15 b)*.

4.3.4 Kansas-City-Standard (KC-Standard, BYTE-Standard)

Am 7./8. November 1975 fand auf Einladung der amerikanischen Mikrocomputer-Zeitschrift BYTE in Kansas City ein Symposium der wichtigsten Mikrocomputerhersteller der USA statt. Dort wurde ein einheitlicher Standard zur Datenaufzeichnung auf handelsüblichen Kassettenrecordern vereinbart [23].

Zugrunde liegt eine Frequenzumtastmethode, die mit den standardisierten Modem-Eckfrequenzen (Modem = Modulator/Demodulator, eine Einheit zur Datenfernübertragung z. B. über Telefonleitungen) von 2400 und 1200 Hz arbeitet *(Bild 16)*. Eine logische 1 besteht aus acht Schwingungen zu 2400 Hz (sog. Mark-Bit), eine logische 0 (Space-Bit) aus vier Schwingungen zu 1200 Hz. Als Datenformat sind 8-bit-Worte gewählt, ohne deren Form auf einen festen Code (z. B. ASCII) festzulegen.

Wie bei asynchronen Verfahren wird die Information von einem Startbit (log. 0) und zwei Stoppbits umrahmt. Da die Länge einer Bitzelle jedoch konstant ist und immer ein ganzzahliges, bekanntes Vielfaches der Taktfrequenz darstellt, handelt es sich zusätzlich um ein synchrones *(self-clocking)* Verfahren. Es ergibt sich so eine Standardübertragungsrate von 300 Bd. 256 Byte werden in 10,5 s übertragen.

Der Standard weist zwar eine nicht zu hohe Redundanz auf, wie der KIM-Standard, ist aber sehr sicher. Im Gegensatz zum KIM-Standard ist er sehr einfach auf höhere Geschwindigkeiten (Speicherdichten) zu verändern. Durch fortgesetztes Halbieren der Bitzelle, bei sonst gleichen Bedingungen, lassen sich Übertragungsraten von 600, 1200 und 2400 Bd erzielen. Praktische Versuche ([18], S. 16) haben jedoch ergeben, daß bei normalen Kassettenrecordern maximal 1200 Bd vertretbar sind. Damit könnte ein Datenblock von 256 Byte in etwa 2,4 s übertragen werden.

4.3.5 Andere Formate

Es werden in der Praxis auch noch andere Formate verwendet. In der Regel beruhen sie auf Frequenzumtastverfahren, wobei jedoch die Eckfrequenzen anders gewählt sind. In der Literatur finden sich Vorschläge von 1270 (log. 1) und 1070 Hz (log. 0) [16] bzw. 2125/2975 Hz [22].

5 Datenformatierung

5.1 Problemstellung

Die bis jetzt vorgestellten Vereinbarungen beziehen sich auf die physische Codierung eines einzelnen Bits oder Bytes. Es gibt nun eine ganze Reihe von Gründen, das Band nicht nur einfach mit einem fortlaufenden Strom von Speicherdaten nach einer der vorgestellten Methoden vollzuschreiben. Dazu gehören:

a) Ein Kassettenrecorder braucht eine „Hochlaufzeit" von mehreren Sekunden, bis der Bandlauf stabil und die Elektronik eingeschwungen ist. Außerdem ist ein Auslaufen des Motors bei den üblichen Start/Stopp-Steuerungen zu berücksichtigen.

b) In der Regel beginnt eine Aufnahme nicht unmittelbar am Bandanfang; ihr Beginn muß sich auch ohne Suchlauf zuverlässig finden lassen.

c) Die Leseelektronik muß die Möglichkeit zur Einsynchronisierung auf die empfangenen Daten haben.

d) Die empfangenen Daten müssen auf Richtigkeit getestet werden können.

e) Bei vielen Programmen ist es wünschenswert, die benötigte Aufnahme vom Computer selbst suchen zu lassen.

Zu a):
Werden mehrere unabhängige Aufnahmen auf einem Band gespeichert, so muß die Lücke zwischen ihnen *(interrecord gap)* groß genug sein, um das Auslaufen und Wiederanlaufen des Motors (auch bei Versetzen der Aufnahme von Hand) zuverlässig vor dem eigentlichen Datenstrom zu gewährleisten. Vor dem Schreiben oder Lesen irgendwelcher Information sollte so eine Verzögerung von 1...3 s eingeschoben werden.

Zu b):
Es gibt einen besonderen Grund, eine Aufnahme nicht am Bandanfang beginnen zu lassen: Trotz Vorspann sind die ersten Bandabschnitte einer Beschädigung und Verschmutzung besonders ausgesetzt, was leicht zu einem Datenverlust führen kann.
Im Kansas-City-Standard wird vorgeschrieben, daß die Aufnahmen frühestens 30 s nach dem Bandanfang beginnen sollen [23]. Es empfiehlt sich sehr, dieser Forderung auch bei anderen Aufzeichnungsmethoden zu folgen.

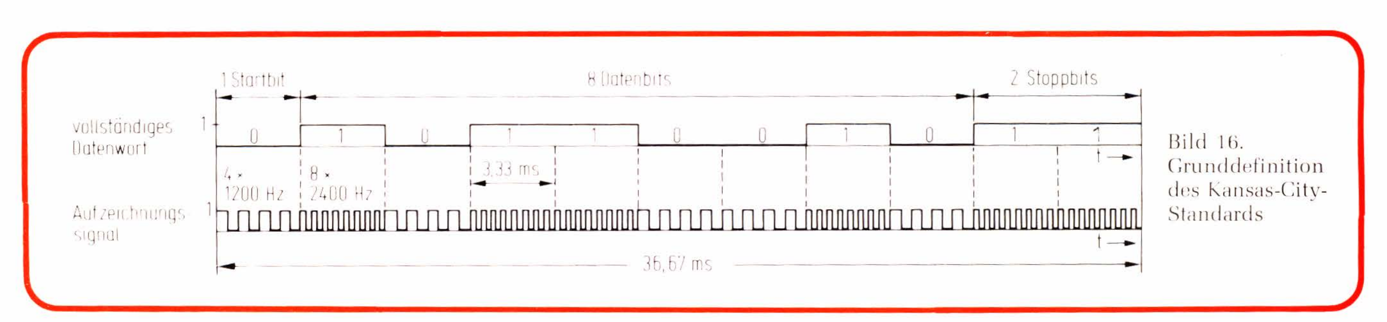


Bild 16. Grunddefinition des Kansas-City-Standards

Zu c)...e):
Diese Punkte sind sehr systembezogen. Auf sie soll im folgenden etwas näher eingegangen werden. Dabei werden nur noch die Formate Tarbell, KIM und Kansas-City betrachtet, für die Formatdefinitionen bzw. -vorschläge vorliegen.

5.2 Systemforderungen

5.2.1 Blocktrennung und Synchronisation

In der Praxis sucht man eine Aufnahme unter mehreren auf dem Band anhand des Bandzählers grob auf und findet ihren genaueren Anfang nach Gehör. Das erfordert eine akustisch erkennbare Blocktrennung. In der Regel ist das die Aufzeichnungslücke zwischen der vorhergehenden und der folgenden Aufnahme.

Von dieser Stelle ab muß der Computer sich auf die Aufnahme einsynchronisieren können. Dazu werden bereitgestellt:

- ein Vorspann von etwa 5s Länge als Puffer für den Hochlauf des Recorders,
- bei den Synchronverfahren eine Möglichkeit, rechtzeitig die Bytegrenzen aus dem Bitstrom aufzufinden und
- ein spezielles Zeichen bzw. eine Zeichenkombination, womit der Beginn der eigentlichen Aufnahme gekennzeichnet wird.

Am einfachsten wird der Kansas-City-Standard diesen Forderungen gerecht *(Bild 17 a)*. Er fordert zu Beginn jeder Aufnahme einen 5 s langen ununterbrochenen Strom von logischen Einsen, d. h. eine kontinuierliche Aufzeichnung von 2400 Hz. Diese ist auch nach Gehör eindeutig von dem folgenden Datenstrom zu unterscheiden. Da es sich um ein byteorientiert-asynchrones Verfahren handelt, liegt der Beginn der Aufnahme durch das Startbit, d. h. die erste auftretende logische 0, eindeutig ohne weitere Vorkehrungen fest.

Die beiden anderen Methoden verfahren in prinzipiell gleicher Weise. Der Vorspann dient neben der Aufnahmetrennung gleichzeitig zur Einsynchronisation auf die Bytegrenzen. Dies geschieht durch Definition eines bestimmten Bitmusters, wodurch der Vorspann aber nicht mehr eindeutig nach Gehör (ohne die vorangehende Aufnahmelücke) lokalisiert werden kann (bei dem nur eine Frequenz verwendenden Tarbell-Verfahren ist dies schon prinzipiell ausgeschlossen).

Das KIM-Format benutzt das im ASCII-Code hierfür vorgesehene Zeichen SYN (SYNchronisation), sedezimal 16. Eine Folge von 100 SYN-Zeichen ergibt so bei 134 Bd einen ca. 6 s dauernden Vorspann. Das Bitmuster weist hierbei an den Bytegrenzen eine Folge von 4 Nullen auf, von denen die erste noch zum vorangehenden Byte gehört *(Bild 17 b)*. Diese Folge kann leicht erkannt und zur Synchronisation verwendet werden.

Das Tarbell-Interface invertiert die ersten 4 bit des SYN-Zeichens und zeichnet so als Vorspann etwa 5 s lang sedezimal E6 auf *(Bild 17 c)*. In diesem Bitmuster ist der Byteanfang eindeutig durch eine Folge von drei logischen Einsen gegeben.

Die Besonderheiten des Phasenumtastverfahrens bringen es mit sich, daß keine Dreiergruppe von log. 0 im Bitstrom vorkommen darf, um die richtige Phasenlage der Aufnahme ermitteln zu können, was beim Austausch von Aufnahmen wichtig wird.

Nach erfolgter Synchronisation kann der Bitstrom auf ein Startzeichen für die eigentliche Aufnahme hin abgesucht werden. Im KIM-Format wird hierfür die ASCII-Codierung eines Sterns, sedezimal 2A, verwendet. Das Tarbell-Interface verwendet sedezimal C3, gefolgt von einem weiteren Synchronisationsbyte E6.

5.2.2 Datenformatierung

5.2.2.1 KIM

Für die nach dem Vorspann folgenden Daten schreibt nur das KIM-Interface ein festes Format vor *(Bild 18)*. Es entspricht weitgehend dem schon besprochenen Lochstreifen-Format. Auch hier wird je-

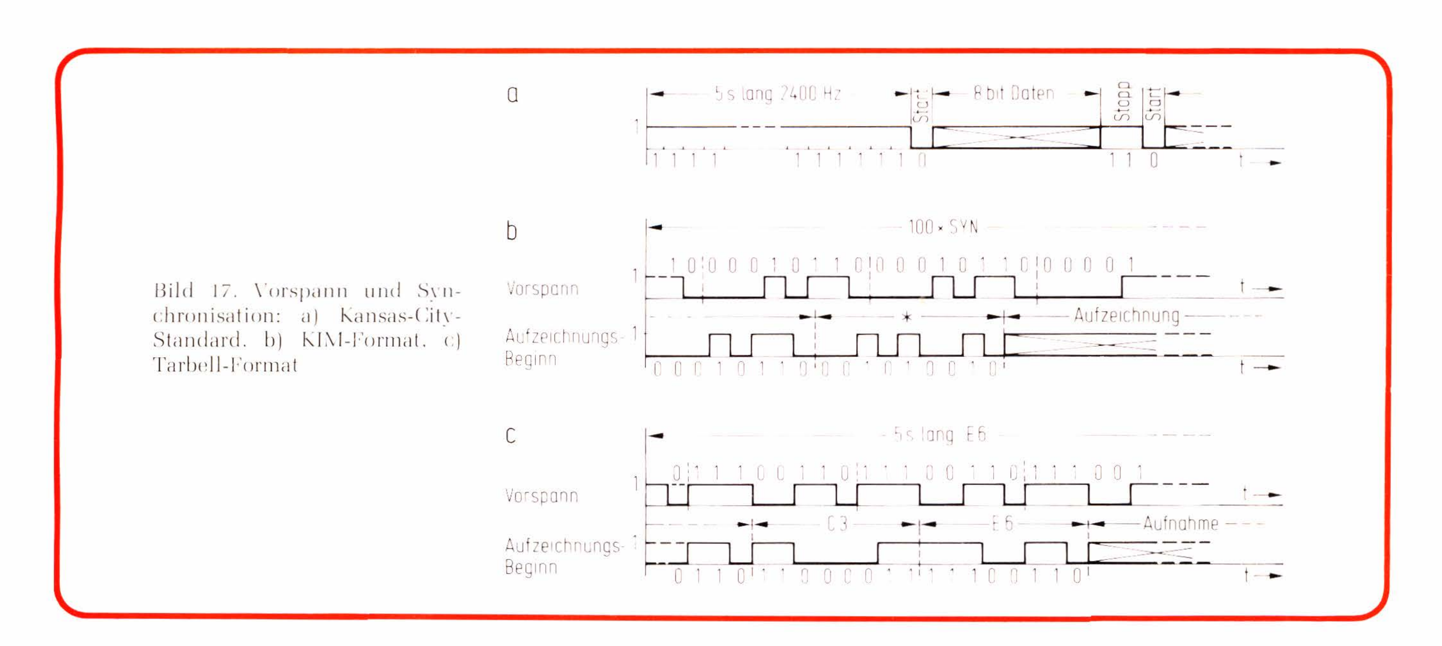


Bild 17. Vorspann und Synchronisation: a) Kansas-City-Standard, b) KIM-Format, c) Tarbell-Format

des Datenbyte in zwei 4-bit-Hälften zerlegt, von denen jede als ASCII 0...F codiert wird. Jedem ASCII-Zeichen wird ein Paritätsbit hinzugefügt. Durch die hohe Redundanz wird zwar die Datensicherheit wesentlich verbessert, die faktische Übertragungsrate aber auf 67 Bd verringert. Als weitere Sicherheit wird über den zu übertragenden Datenblock eine 16 bit breite Quersumme gebildet *(checksum)* und (uncodiert) mit dem niederwertigen Byte voran nach dem Datenblock auf das Band geschrieben (CKL, CKH). Daten und Prüfsumme werden durch das ASCII-Zeichen „/" voneinander getrennt, die Aufnahme selbst endet mit zwei EOT-Zeichen *(End-of-Tape*, Bandende), sedezimal 04.

Der Datenblock wiederum wird eingeleitet von einer Identifizierungsnummer (ID) 0...255 und der Startadresse, an der die Daten (normalerweise) im Speicher einzulesen sind (ebenfalls uncodiert, niederwertiges Byte voran (SAL, SAH). Die ID-Nummer gestattet das programmgesteuerte Aufsuchen einer bestimmten Aufnahme.

5.2.2.2 Andere Vorschläge

Für die beiden anderen Aufzeichnungsverfahren gibt es keine derart verbindlichen Konventionen zum Datenformat. Im Unterschied zum KIM-Format sind sie beide nicht auf den ASCII-Code festgelegt.

Der Tarbell-Standard kennt weder hinsichtlich der Länge eines Datenworts noch dessen Codierung eine Einschränkung. Der Kansas-City-Standard ist durch sein spezielles asynchrones Format auf 8-bit-Worte festgelegt, deren Inhalt keinen Einschränkungen unterworfen ist. Für kürzere Wortlängen ist vereinbart, sie unmittelbar nach dem Startbit beginnen zu lassen und den Rest mit Stoppbits (log. 1) aufzufüllen. Längere Wortlängen lassen sich so durch Zusammensetzen mehrerer Einzelworte übertragen.

Weitere Vorschläge zur Formatierung der Datenblöcke liegen in [17], [30] und [31] vor. Im wesentlichen geht es dabei darum, die Aufnahmen in logisch zusammengehörige Abschnitte *(Files)* zu gliedern, die selbst unter Umständen in Unterabschnitte *(blocks)*

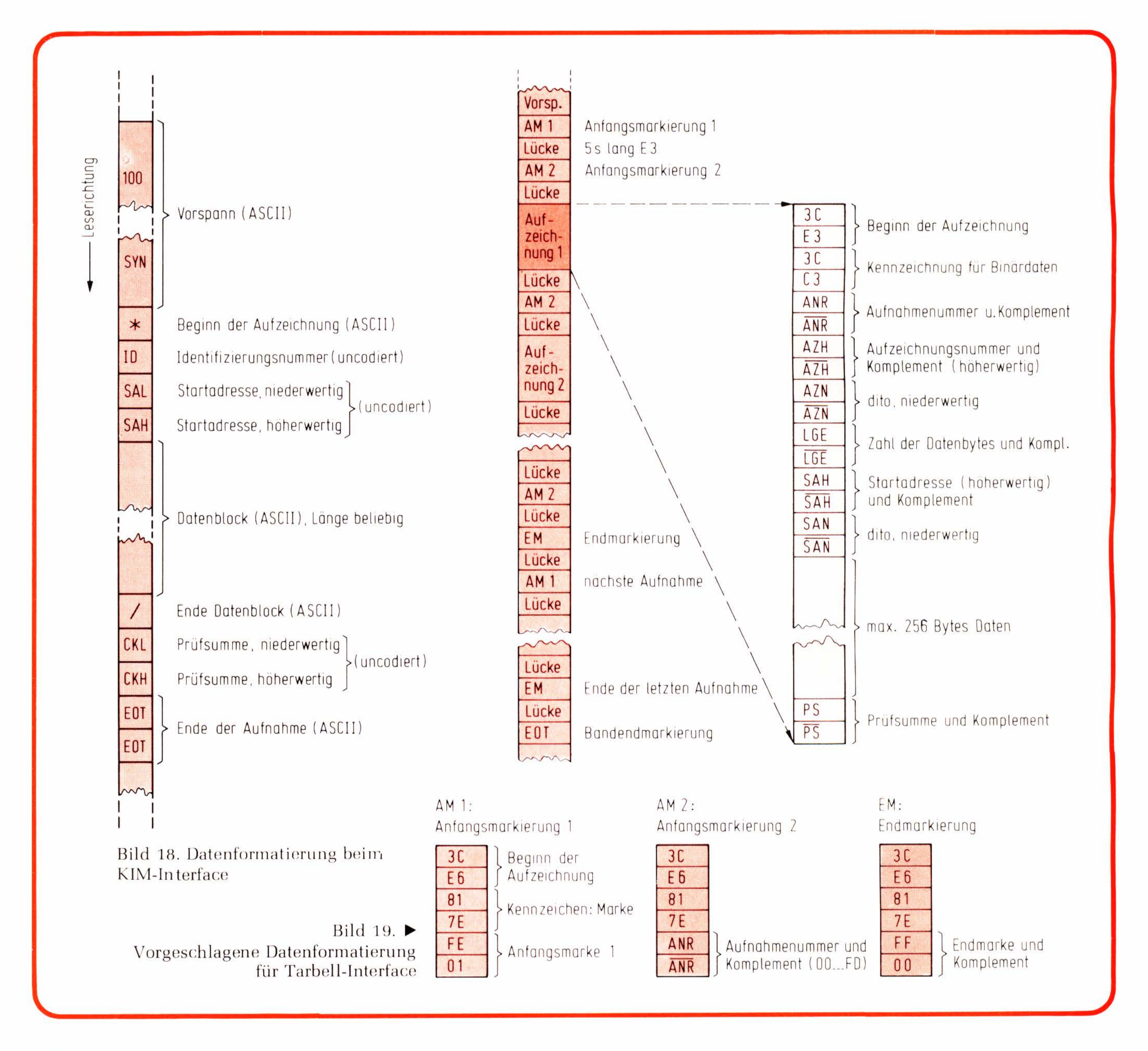


Bild 18. Datenformatierung beim KIM-Interface

Bild 19. ▶ Vorgeschlagene Datenformatierung für Tarbell-Interface

festgelegter oder angegebener Länge eventuell mit Prüfsumme zerlegt sind, Startadressen und Blocklängen festzuhalten und bestimmte Plätze für die Prüfsumme vorzusehen.

|17| ist auf das Tarbell-Format zugeschnitten und teilt den Datenstrom in Blöcke zu maximal je 256 Byte auf, denen Anfangsmarkierungen (*file marks*, mit passenden Aufnahmelücken dazwischen) vorangehen, gefolgt von den erwähnten Informationen über die Aufnahme, den Daten selbst, dann der nächsten Anfangsmarkierung usw. bis alle Daten gespeichert sind (*Bild 19*). Den Aufnahmen als Gesamtes wird eine besondere Anfangsmarkierung vorangestellt, abgeschlossen werden sie durch eine besondere Endmarkierung.

In |31| werden die besonderen Eigenschaften des Interfaces für den Kansas-City-Standard „Speakeasy" von Morrow's Micro Stuff |30|, das als Universal-Interface weit verbreitet ist, ausgenutzt. Dieses Interface ist voll softwareorientiert (es kann auch für die anderen vorgestellten Standards programmiert werden) und benutzt für den Kansas-City-Standard mit 300 Bd einen im ROM auf der Karte (S-100-Bus) abgelegten Urlader (*Bootstrap*), der ohne weiteren Aufwand 256 Byte in das auf der Karte befindliche RAM einliest und dann zu dessen erstem Byte springt.

Der Assembler und Text-Editor ATE |31| benutzt diese Eigenschaft, indem er jeder Aufnahme einen 256 Byte langen Vorspann voranstellt, der außer den Informationen über Titel der Aufzeichnung, deren Art (Text oder Objektcode), Startadresse, Blocklänge und Prüfsumme seinerseits die nötigen Laderoutinen enthält, um die Aufzeichnung auch unabhängig von ATE automatisch laden und auf Richtigkeit prüfen zu können, wobei noch die im Speakeasy-ROM abgelegte Routine zur Prüfsummenberechnung benutzt wird. Jede Aufnahme besteht so aus zwei getrennten Aufzeichnungen: dem 256 Byte langen Vorspann und dem eigentlichen Datenblock, dessen Länge im verfügbaren 64-K-Adreßraum keine Grenze gesetzt ist.

6 Diskussion

Es spricht vieles dafür, daß sich in Zukunft der Kansas-City-Standard in Verbindung mit Kassettenrecordern der unteren Mittelklasse als Aufzeichnungsstandard durchsetzen wird. Die beiden anderen Standards sind – obgleich weit verbreitet – zu firmenspezifisch. KIM hat darüber hinaus doch zuviel Redundanz in der Definition, um wirtschaftlich bei größeren Datenmengen eingesetzt zu werden. Der Kansas-City-Standard ist andererseits etwas zuverlässiger als Tarbell und nicht so sehr von dem verwendeten Recorder abhängig. Vor allem bietet er durch sein asynchrones Format die Möglichkeit, die Daten nicht in einem kontinuierlichen Strom, sondern Byte für Byte aufzuzeichnen, was manchmal von Vorteil sein kann. Diese Möglichkeit hat das synchrone Tarbell-Format prinzipiell nicht.

Was noch aussteht, ist eine (oder – den Anwendungsfällen angepaßt – mehrere) verbindliche Definition des Datenformats, was den Programmaustausch wesentlich erleichtern würde. Es sollte Urladerfähigkeiten ähnlich ATE besitzen und möglicherweise (der leichteren Handhabung wegen) eine Blockstruktur ähnlich |17| aufweisen. Auf diese Weise könnten sich die Massenspeichermethoden mit Kassettenrecordern bei gegebenen Anforderungen und Geschwindigkeitsbeschränkungen vereinheitlichen und ihr Einsatzbereich wesentlich erweitern lassen.

Literatur

|1| *Timm, V.*: Im Blickpunkt: ROMs, PROMs und PLAs. Technologien – Arten – Programmierung. ELEKTRONIK 1976, H. 5, S. 38...47.

|2| *Müller, R.*: Prinzip und Arbeitsweise von Floppy-Disk-Speichern. ELEKTRONIK 1978, H. 6, S. 57...62.

|2a| *Rampil, J.*: A Floppy Disk Tutorial. BYTE 1977, H. 12, S. 24...44.

|3| *Budnick, K.*: Bar Code Loader. BITS Inc., 70 Main Street, Peterborough N. H. 03458. Zitiert nach: BYTE 1978, H. 4, S. 137.

|4| *Mosely, R. C.*: A Low Cost Light Wand Amplifier. BYTE 1978, H. 5, S. 92...95

|5| M 6800 Microprocessor Applications Manual. Motorola Inc., 1975

|6| Codes für die Daten- und Nachrichtentechnik. ELEKTRONIK Arbeitsblatt Nr. 101 (Datentechnik). ELEKTRONIK 1977, H. 1, S. 81...82 und H. 2, S. 79...80.

|7| Introduction to Programming. Maynard, Mass.: Digital Equipment Corporation 4/1973. PDP-8 Handbook Series.

|8| Intersil IM 6100 CMOS 12 Bit Microprocessor. Cupertino: Intersil 1976. Datenblatt- und Applikationssammlung.

|9| *Scheil, O.*: Selbstbau-Mikrocomputer mit CMOS-Bausteinen. ELEKTRONIK 1978, H. 4, S. 79...84.

|10| MCS 85 User's Manual. Santa Clara, Cal.: Intel Corp., March 1977. Handbuch 8085-Prozessor mit Datenblatt- und Applikationssammlung.

|11| KIM 1 Handbuch (deutsch). Friedrichsdorf: MOS Technology Inc., 3/1977.

|12| *Lesea/Zaks*: Microprocessor Interfacing Techniques. Berkeley – Paris: Sybex, 2/1978

|13| *Freihoff, W.*: Digitale Rechenanlagen. Pforzheim: ITT, 3/1975. (Lehrgang Digital-Elektronik, Heft 9).

|14| *Gößler, R.*: Einführung in die Mikrocomputer-Programmierung (V). ELEKTRONIK 1977, H. 9, S. 71...78.

|15| Software auf Schallplatten. Elektor 89, Mai 1978, S. 5–29.

|16| *Staub, H. J.*: Bandspeicher. micomp 1977, H. 1, S. 45 ff.

|17| *McDonough* and *Hammontre*: Cassette I/O Format... Standards are still needed! Kilobaud 1977, H. 8, S. 18...23.

|18| The „Kill a Byte" Standard – Revisited. Kilobaud 1977, H. 5, S. 16 ff und 21 ff. (Diskussion zu Tarbell- und Kansas-City-Standard).

|19| *Hughes, P.*: Make Your Investment Count...the inside view of a custom MP-68. Kilobaud 1977, H. 5, S. 38...44.

|20| *Clarke, S.*: A Home Computer Pioneer...profile of Don Tarbell. Kilobaud 1977, H. 5, S. 132...138.

|21| *Mohler, L. S.*: A Clean Cassette...getting the most from inexpensive recorders. Kilobaud 1977, H. 6, S. 76 ff.

|22| *Bourdeau, D. R.*: Cassette Interface First Aid...use your processor to set timing. Kilobaud 1977, H. 7, S. 49.

|23| *Peschke, M.* and *V.*: Report: BYTE's Audio Cassette Standards Symposium. Byte 1976, H. 2, S. 72 ff.

|24| *Gray, K.*: The Designer's Eye View of the AC-30. Byte 1976, H. 16, S. 98...108.

|25| Datapack Cassette Digital Magnetic Tape Memory System. TEAC MT-2. Datenblatt der TEAC Corp., 1978. Zu beziehen über Byte-Shop Logitec, München.

|26| The Tarbell Cassette Interface. Carson Cal.: Tarbell Electronics 1977. (Anwenderhandbuch)

|27| *Beifuß, W.*: Digitale Phasensynchronisiereinheit für synchrone Datenübertragungen. ELEKTRONIK 1978, H. 2, S. 52...56.

|28| *Fronheiser, K.*: Asynchrone Mikrocomputerschnittstelle. ELEKTRONIK 1978, H. 1, S. 45...50.

|29| Datamega Katalog: Drucker, Kassetten- und Floppy-Laufwerke. Ausgabe 1.10.77. Datamega KG, Putzbrunn/München.

|30| Bauanleitung und Dokumentation zum Speakeasy I/O-Board von Morrow's Micro Stuff (Universelles I/O Interface für den S-100-Bus). Zu beziehen über Computershop GmbH, Freiburg.

|31| Handbuch ATE – Assembler & Text-Editor für 8080-Systeme von Gary Fitts. Zu beziehen über Computershop GmbH, Freiburg.

|32| *Schulz, A.*: Informatik für Anwender. de Gruyter, Berlin, New York.



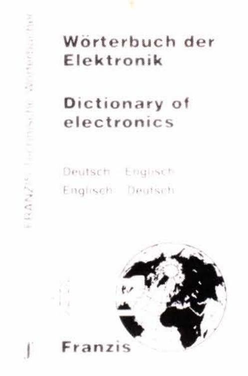
Franzis-Verlag München

der große Fachverlag
für angewandte Elektronik.



Wörterbuch der Elektronik Deutsch-Englisch Englisch-Deutsch

Von Reinhard Krönert.



Dieses Spezialwörterbuch spiegelt den Elektronik-Wortschatz der deutschen und englischen Sprache wieder. Neben den klassischen Elektronik-Begriffen, die vielleicht auch in einem allgemeinen technischen Wörterbuch verzeichnet sind, findet der Benutzer hier jene speziellen Begriffe, die in den letzten zwei, drei Jahren aufgetaucht sind. Es sind jene Elektronik-Wörter, die sich vom philologischen Standpunkt nur schwer in die jeweilige Sprache einordnen lassen und die schlicht Jargon genannt werden. Doch gerade auf deren Übersetzung kommt es an, wenn man sich mit dem Kollegen auf der anderen Seite der Sprachbarriere verständigen will.

Neuerscheinung. 108 Seiten, kart. DM 11.80
ISBN 3-7723-6561-2

Taschen-Tabelle Integrierte Schaltungen – TIS digital

Von Heinrich Müller.



Mehr als 2200 verschiedene Typen digitaler integrierter Schaltungen sind in dieser Tabelle mit ihren wichtigsten Kenndaten erfaßt.

Von großer Bedeutung sind für den Benutzer auch die ausführlichen Sockelschaltungen. Sie sind für jede IS vorhanden, in vielen Fällen gleich mit der Wahrheitstabelle.

Neben Gattern, Flip-Flops, Speichern und Schieberegistern wurden auch alle wichtigen Zähler, Dekoder, Multiplexer, arithmetische Bausteine und Monoflops aufgenommen. Das „Bonbon" dieser Tabelle sind 200 Typen verschiedener Leuchtanzeigen aller Technologien. (Leuchtanzeigen aller Größen und Farben, mit und ohne eingebauter Logik sowie Anzeigeröhren und Flüssigkristalldisplays.)

Neuerscheinung. 496 Seiten, kart. DM 28.–
ISBN 3-7723-6401-2

Einführung in die digitale Steuerungstechnik

Schaltalgebra, Bausteine und Steuerungsprobleme im Selbstunterricht und in der Praxis.

Von H. J. Siegfried.



Was will man mehr! Logische Verknüpfungsschaltungen können ohne elektronische Vorkenntnisse aufgebaut werden. Die Logik kommt mit den Schaltzeichen der digitalen Informationsverarbeitung aus. So lassen sich auch Flip-Flops erklären, aus denen Schieberegister und Zähler zusammengesetzt werden. Leicht verständlich und folgerichtig wird die Entwicklung vom einfachen zum schweren Sachverhalt dargestellt.

Das Buch entstand aus dem Unterricht an Berufsfachschulen und Technikerschulen. Praxisnahe Beispiele, Merksätze, Übungs- und Wiederholungsaufgaben vermitteln die Grundkenntnisse, die zum Verständnis der modernen digitalen Steuerungstechnik unerläßlich sind.

Neuerscheinung. 169 Seiten, 169 Abbildungen und zahlreiche Tabellen. Lwstr-kart. DM 30.–
ISBN 3-7723-6411-X

Praxis mit Mikroprozessoren

Wie herkömmliche Digitalschaltungen durch Mikroprozessoren erweitert, ausgebaut oder ersetzt werden können.

Von Horst Pelka.



Wer versteht schon Mikroprozessoren richtig anzuwenden? Wer weiß sie richtig zu nutzen? Hier ist das praxisnahe Buch, das im Experiment den Mikroprozessor in seiner Alltäglichkeit zeigt.

Der Band beschreibt ein Mikrocomputersystem, das modular aufgebaut ist und das nachgebaut werden kann. Der modulare Aufbau gewährleistet eine nahezu beliebige Erweiterbarkeit und ist an keine bestimmte Type gebunden.

Als Prototyp nimmt der Autor ein System aus der Familie 8080, weil dies der am häufigsten verwendete Typ ist. Es werden zunächst einfache Anwendungen beschrieben, natürlich auch Hardware, Peripherie, Zusatz-, Hilfs- und Prüfschaltungen, um bei der Software und der PRIM-Programmier-Einrichtung zu enden.

„Jetzt habe ich es verstanden!" wird zu sich selber sagen, wer diesen Band in allen Einzelheiten nachvollzogen hat.

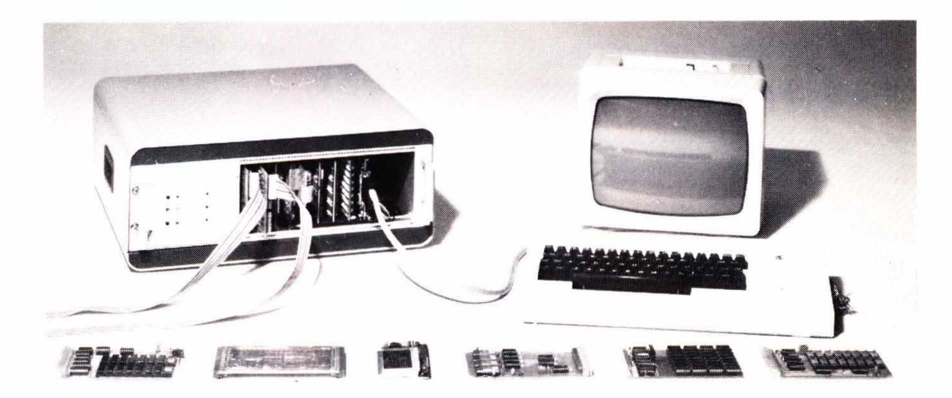
Neuerscheinung. 192 Seiten, 84 Abbildungen.
Lwstr-kart. DM 19.80
ISBN 3-7723-6581-7

Franzis-Bücher erhalten Sie durch jede Buchhandlung, sowie in den einschlägigen Fachhandlungen. Bestellungen auch an den Verlag

ELZET 80

Das wirklich universelle Mikrocomputersystem, auf das Sie schon lange gewartet haben. Aufbaubar in kleinen Schritten mit genormten Teilen

Das ELZET 80-System – Im Europaformat
Alle Systemplatinen (Außer den externen Eingabeschnittstellen HEX-I/O und ASCII-Tastatur) sind konsequent im Europakartenformat 100x160mm aufgebaut. Der Vorteil: Kein unhandlicher Einplatinencomputer, der schlecht in ein Gehäuse paßt, sondern große Packungsdichte, im 19"-Gehäuse sauber verpackt.
Zur größtmöglichen Flexibilität sind alle Systemplatinen voll gepuffert – das kostet zwar etwas mehr, bringt aber wirklich volle Ausbaubarkeit.



Die Busplatine – Rückgrat des Vielseitigen
Stellt mit 20 Steckerplätzen reichlich Platz für beliebige Erweiterungen IHRES Systems zur Verfügung. Jeder benutzte Steckerplatz wird mit einer 64-poligen VG-Buchse nach DIN 41612 bestückt und nimmt jede beliebige Systemkarte an jedem beliebigen Steckerplatz auf. Sollten 20 Steckerplätze nicht ausreichen, kann das System mit einer weiteren Busplatine auf 40 Steckerplätze erweitert werden – so weit reicht die Pufferung (natürlich auch für ,eigene' Karten, die mit LS-Last gepuffert sind). Die Belegung der 64 Pole entspricht dem Kontron-Bus.
Preis: **58,– DM** 64,96 DM

Die CPU – nur eine Nebensache ?
Sie nimmt den geringsten Raum in einem Mikrocomputer ein – leistet dafür aber die ganze Arbeit, die Datenverarbeitung und den Datentransport.
Wir haben uns für die Z80-CPU entschieden, weil sie einen sehr umfangreichen und effizienten Befehlssatz hat und mit 2,5 MHz recht schnell ist. Der im Vergleich zu anderen Fabrikaten höhere Preis geht in die Systemkosten nur zu einem sehr geringen Teil ein. Die CPU-Platine enthält die eigentliche CPU mit einem quarzstabilisierten Clock-Oszillator. Ferner enthält die Platine den bereits verdrahteten Platz für einen USART, also seriellen Ein- und Ausgang zur Außenwelt. Ein besonderer Einfall unserer Techniker ist die Option eines 'Power-On-Jump' – über PROM's können nach RESET wahlweise 64 Speicherplätze automatisch angesprungen werden, in denen z.B. ein Monitorprogramm oder Ihr BASIC steht, anstatt den wertvollen Speicherplatz der Seite 00 (für Interrupts wichtig) belegen zu müssen.
Z80- CPU-Karte
Preis: **178,-- DM** 199,36 DM
Option 'Power-On-Jump'
Preis: **25,– DM** 28,– DM
Serienschnittstelle 20mA
Preis: **32,90 DM** 36,84 DM

Speicher – ein Thema mit Variationen
Ohne Speicher geht praktisch nichts, denn irgendwoher muß die CPU ihre Befehle bekommen. Um die vielfältigen Wünsche aller Kunden befriedigen zu können, stehen für das ELZET80 System mehrere Versionen von Speicherkarten zur Verfügung:

1. Die Monitorplatine
Bestückbar mit insgesamt 8 kByte Speicher, der beliebig (innerhalb voller kByte-Grenzen) als stat. RAM oder EPROM ausführbar ist.
4/79 auf Anfrage
2. Die 4K RAM-Platine
Eine besonders preisgünstige Platine, die mit statischen RAM's 21L02 aufgebaut ist.
Preis: **250,– DM** 280,– DM
3. Die 8K RAM-Platine
Bestückt mit stat. RAM's TMS4044 4Kx1 Stromverbrauch nur 1,2 A. Auch in 4 KB (halb bestückt) lieferbar.
8K-Karte, 4K bestückt
Preis: **428,-- DM** 479,36 DM
8K-Karte, voll bestückt
Preis: **659,90 DM** 739,09 DM

4. Die 32K RAM-Platine (dyn.)
Für den großen Bedarf an Halbleiterspeicher, bestückt mit 16 dyn. RAM's 4116 (16Kx1), lieferbar ca. 3/79
Preis: **698,– DM** 781,76 DM
5. Die 4K EPROM-Platine
Für EPROM's Typ 2708, bestückbar in 4 Schritten zu je 1 kByte. Für Monitorprogramm oder z.B. BASIC-Interpreter. Bestückt mit 1x 2708.
Preis: **131,80 DM** 147,62 DM
6. Die 2K CMOS-RAM-Karte
Mit stat. CMOS-RAM's 5101, 450ns. Die Karte ist mit einem Akku gepuffert, damit halten die RAM's auch bei Abschalten des Systems die Information.
2K CMOS-RAM, 1/4K bestückt
Preis: **249,– DM** 278,88 DM
2K CMOS-RAM, voll bestückt
Preis: **479,– DM** 536,48 DM
Alle Speicher-IC's werden mit Fassungen geliefert, alle Platinen sind adreßdekodiert und voll gepuffert.

HEX-Input/Output
Programmieren ohne Aufwand
Diese Platine ist bestückt mit einer Hexadezimaltastatur und Kommandotasten und ist über Kabel direkt mit dem Systembus verbunden. Sie erlaubt es, Programme direkt in HEX in den Speicher einzugeben und im Einzelschrittverfahren zu testen. Dabei wird kein Monitorprogramm benötigt, die Tasten- und Anzeigendekodierung erfolgt hardwaremäßig. Wertvoller Speicherplatz bleibt so frei und – fast noch wichtiger: Das System bleibt anpassungsfähig da kein Monitorprogramm bestimmte Speicherplätze fest belegt.
Die HEX-I/O ist vor allem für den Anfänger geeignet, der sich mit seiner CPU vertraut machen möchte, aber auch für den Anwender, der Steuerungen programmieren möchte (z.B. Modellbau oder Prozeßsteuerung). Alle Befehle werden direkt in Maschinensprache eingegeben, sicher nicht so einfach wie BASIC, aber man hat einen besseren Einblick in die Arbeit der CPU und der Aufwand ist wesentlich geringer! (Kein ASCII-Keyboard nötig, kein Video-Interface und kein BASIC-Programm.)
Preis: **133,50 DM** 149,52 DM

Kansas-City gibt Sicherheit
Übertragung Mikroprozessor – Cassette kann sehr einfach sein, indem man nämlich aus '1' einen Ton und aus '0' keinen Ton macht. Das hat unter anderem den Nachteil, daß schon bei geringen Geschwindigkeitsunterschieden zwischen zwei Rekordern keine Austauschbarkeit gewährleistet ist. Deshalb wurde auf einer Tagung in Kansas-City ein Verfahren genormt, welches dem frequenzmodulierten Signal auch die Clockimpulse beimischt. So kann die PLL auf der Karte große Geschwindigkeitsunterschiede fehlerlos verarbeiten. Nur schade, daß das KC-Verfahren bisher nur von wenigen grossen Systemen verwendet wird – aber es ist ja auch nicht ganz billig...
KC-Cassetten-Interface
Preis: **129,– DM** 144,48 DM

Aber auch BASIC ist möglich
Und das gleich in drei Versionen: Als Tiny-BASIC in Deutsch (PROFAN), das in 2K EPROM steckt und als Industrie-BASIC in Englisch in 8K oder 12K EPROM. Damit können Sie dann wirklich alles machen, benötigen jedoch dazu:

Die ASCII-Tastatur
mit professionellem, vergossenem Tastenfeld (63 Tasten) und vollem ASCII-Zeichensatz, Tasten elektronisch entprellt.
Preis: **173,– DM** 193,76 DM

Das Video-Interface
Völlig eigenständige Karte mit eigenem 1K-ASCII-Speicher. 64 Zeichen, 16 Zeilen. Seriell (110 - 1200 Baud) und Parallelbetrieb. Anschluß von Tastatur und Serienschnittstelle über 25-poligen D-Stecker. BAS-Ausgang ist für deutsche Norm ausgelegt.
Preis: **399,50 DM** 447,44 DM

Und was man sonst noch so braucht

19"-Baugruppenträger	**79,90 DM**	89,49 DM
19"-Luxusgehäuse	**124,– DM**	138,88 DM
VG 64 a+c, Stecker	**6,50 DM**	7,28 DM
VG 64 a+c, Buchse	**10,50 DM**	11,76 DM

Weiteres Zubehör auf Anfrage.

Zwei besonders günstige Komplettangebote :
1. ELZET 80 Minimalsystem
Baugruppenträger, Busplatine, CPU-Karte, 4K-RAM-Karte mit 1K bestückt, HEX-I/O, Netzteil, Systemhandbuch. Der ideale Einstieg, für nur **598,– DM** 669,76 DM
2. ELZET 80 ASCII-System
Busplatine, CPU-Karte, 4K RAM voll bestückt, Serienschnittstelle, Monitor (für Video, Keyboard und Cass.-Interface) auf 1K einer 4K-EPROM-Karte, Tastatur, Video-Display-Karte, KC Cassetten-Interface und Handbuch
Systempreis: **1198,– DM** 1341,76 DM

Und alle Unterlagen in Deutsch !

Alle fettgedruckten Preise excl. MwSt., alle normal gedruckten incl. MwSt. Angebot und Liefermöglichkeit freibleibend.
Wir beraten Sie gerne ausführlich telefonisch.

ELEKTRONIKLADEN
DETMOLD MÜNSTER AMSTERDAM PARIS
Wilhelm-Mellies-Str. 88
4930 Detmold 18
Telefon (05232) 81 31 , Telex 0931473 laden d



der große Fachverlag
für angewandte Elektronik.

Franzis-Verlag München



Analoge integrierte Schaltungen



Ein Lehrbuch, Schaltungen mit Operationsverstärkern u. analogen Multiplizierern zu entwerfen.

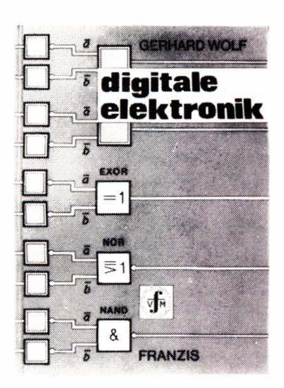
Von Dipl.-Ing. Miklós Herpy.

Um mit dem Operationsverstärker wie mit einem Transistor umgehen zu können, braucht der Elektroniker dieses Lehr- und Applikationsbuch. Die Verbindung zu Bekanntem herstellend, zerlegt der Autor die integrierten Analogschaltungen in Funktionseinheiten, wie das aus der allgemeinen Transistor-Schaltungstechnik bekannt ist. Dieser Vorgang ist sehr ausführlich und auch formelmäßig, d. h., mathematisch erfaßt. Wer die einzelnen Stufen verstanden hat, kann mit einem Operationsverstärker so umgehen wie mit einem Transistor und – was genauso wichtig ist – er kann auf einen Blick die Unterschiede der auf dem Markt so zahlreich angebotenen Typen feststellen. Um dies noch intensiver zu unterstützen, sind in dem Werk alle wichtigen Standard-Operationsverstärker beschrieben.

522 Seiten, 373 Abbildungen, Leinen geb. DM 68.–

ISBN 3-7723-6151-X

Digitale Elektronik



Die Arbeitsweise von integrierten Logik- und Speicher-Elementen.

Von Gerhard Wolf.

Der in der Digitaltechnik tätige Ingenieur erhält mit diesem Buch einen Überblick über den neuesten Stand der Digital-Technik, der ihn in die Lage setzt, nicht nur Baustein-, Speicher- und Schaltkreis-Probleme zu lösen, sondern auch Aufgaben rein logischer Natur, so die Verknüpfung oder die Umwandlung von Informationen im Sinne mathematischer Zusammenhänge, zu bewältigen.

In dieser nunmehr 4. Auflage sind die Abschnitte über Speichertechnik und Schieberegister auf den neuesten Stand der Bipolar- und MOS-Technik gebracht, und die Abschnitte über Leistungskopplung- und Reflexionen mit neuem Material ergänzt.

Diese solide und trotzdem leicht verständliche theoretische Grundlage ergibt ein Know-how, das sich erfolgreich in der Praxis anwenden läßt.

221 Seiten mit 156 Abbildungen und zahlreichen Tabellen. Lwstr-geb. DM 48.–

ISBN 3-7723-5574-9

Industrielle Elektronik-Schaltungen



Eine praxisnahe Schaltungssammlung aus der professionellen Elektronik für Analog- und Digital-Techniker.

Von G. Klasche und R. Hofer.

Etwa 18 Pfennige kostet den Leser jede Schaltung, die in diesem Band aufgeführt, beschrieben und erörtert ist. Wie hoch wäre für ihn der Aufwand in DM, wenn er sie aus dem Nichts entwickeln müßte?

Diese Schaltungssammlung umfaßt die breite Palette der Industrie-Elektronik mit dem Schwergewicht Digitaltechnik. Frei von irgendwelchen Firmenbindungen kam es den Herausgebern darauf an, Schaltungen mit dem gewissen Etwas, mit dem Kniff zu suchen, zu finden und herauszubringen. So unterscheiden sie sich wohltuend von den üblichen Allerweltsschaltungen.

Aus dem Inhalt: Allgemeine Digitalschaltungen. Interfaceschaltungen. Oszillatoren und Generatoren. Operationsverstärkerschaltungen. Steuer- und Regelschaltungen. Filter- und Rechenschaltungen. Meß- und Prüfschaltungen. Optoelektronische Schaltungen. Stromversorgungsschaltungen. Spezialschaltungen. Hobbyschaltungen. Neuerscheinung.

336 Seiten, 176 Abbildungen. Lwstr-geb. DM 38.–

ISBN 3-7723-6441-1

Mikrocomputersysteme



Selbstbau, Programmierung, Anwendung.

Von Rolf-Dieter Klein.

Kaum zu glauben, daß ein Mikrocomputer im Selbstbau hergestellt werden kann! Zunächst muß die Hardware geschaffen werden. Eingabetastatur, Mikroprozessor, Speicher verschiedener Art, Drucker, Sichtgerät, das alles muß zu einer funktionierenden Einheit zusammengeschlossen werden. Und das geht, sogar mit preiswerten, modernen Teilen, die in den Fachhandlungen zu haben sind. – Nun die Software. Da zeigt der Autor mehrere Möglichkeiten auf. Nicht etwa nur ein kleines Programm. Nein, ausführliche Programme werden vorgestellt, die zahlreiche Spiele, mathematische Aufgaben, wissenschaftliche Probleme bearbeiten können. – Als Abschluß und Höhepunkt fügt der Autor Anregungen hinzu, selbst Programme zu schreiben und in dem eigenen Mikrocomputer zu erproben. Was will man mehr?

2., verbesserte Auflage. 159 Seiten mit 133 Abbildungen und 11 Tabellen.

Lwstr-geb. DM 29.–.

ISBN 3-7723-6382-2

Franzis-Bücher erhalten Sie durch jede Buchhandlung, sowie in den einschlägigen Fachhandlungen. Bestellungen auch an den Verlag